UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

AD-A197 149

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/CI/NR 88-165
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER: DTIC FILE COPY
4. TITLE *(and Subtitle)*: HYPERVELOCITY ORBITAL INTERCEPT GUIDANCE
5. TYPE OF REPORT & PERIOD COVERED: X THESIS
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): SALVATORE ALFANO
8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT STUDENT AT: UNIVERSITY OF COLORADO
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:
12. REPORT DATE: 1988
13. NUMBER OF PAGES: 184
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: AFIT/NR, Wright-Patterson AFB OH 45433-6583
15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT *(of this Report)*: DISTRIBUTED UNLIMITED: APPROVED FOR PUBLIC RELEASE
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*: SAME AS REPORT
18. SUPPLEMENTARY NOTES: Approved for Public Release: IAW AFR 190-1
LYNN E. WOLAVER 8 Aug 88
Dean for Research and Professional Development
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*: ATTACHED

DTIC ELECTE AUG 17 1988 S D H

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

88 8 16 033

HYPERVELOCITY ORBITAL INTERCEPT GUIDANCE

by

SALVATORE ALFANO

B.S., United States Air Force Academy, 1974

M.S., Air Force Institute of Technology, 1982

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical Engineering

1988

This thesis for the Doctor of Philosophy degree by

Salvatore Alfano

has been approved for the

Department of

Electrical Engineering

by

Charles E. Fosha, Jr.

John M. Liebetreu

Date 14 April 1988

To my wife, Michele, for her unfailing love and support

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for giving man the ability, limited though it may be, to see and understand the workings of His wisdom and His hand in all of creation. My thanks go to Dr. Charles E. Fosha, Jr., my thesis advisor, and Dr. Robert B. Asher, for their guidance on this project. I would also like to thank Dr. John M. Liebetreu for his careful review of this manuscript as a second reader. In addition, I wish to thank the other members of my examining committee, Drs. Saber Elaydi, Ronald M. Sega, Renjeng Su, Mark A. Wickert, and Rodger E. Ziemer for their helpful suggestions.

Alfano, Salvatore (Ph.D., Electrical Engineering)

Hypervelocity Orbital Intercept Guidance

Thesis directed by Associate Professor Charles E. Fosha, Jr.

Terminal guidance of a hypervelocity exo-atmospheric orbital interceptor with free end-time is examined. The pursuer is constrained to lateral thrusting with the evader modeled as an ICBM in its final boost phase. Proportional navigation, optimal control using certainty equivalence, dual control, and control with optimum thrust spacing are all examined. Also, a new approach called certainty control is developed for this problem. This algorithm constrains the final state to a function of projected estimate error to reduce control energy expenditure. All methods model the trajectories using splines and employ eight state Extended Kalman Filters with line-of-sight and range updates. The relative effectiveness of these control strategies is illustrated by applying them to various intercept problems.

CONTENTS

TABLES

Table

FIGURES

CHAPTER I

INTRODUCTION

Orbital interceptor performance can be enhanced by using a terminal guidance law that incorporates the orbital dynamics of the pursuer and evader plus the error knowledge of their estimates. The purpose of this research is to develop a guidance scheme for a hypervelocity, exo-atmospheric orbital vehicle in the final thirty seconds of flight that minimizes lateral thrusting while attempting to intercept a boosting missile. Much work has been done on the most common form of intercept guidance, proportional navigation, and its variations. This type of navigation assumes that the force of gravity acts equally on the pursuer and evader and can therefore be ignored in the relative dynamics. For orbital intercepts with large initial ranges the force of gravity will affect the relative trajectory and should be included in the equations of motion. To date, analytic solutions for such intercepts exist only when the pursuer's impact conditions are prespecified.

The general guidance schemes studied in this research attempt to minimize lateral velocity changes by varying the impact conditions through the use of splines. The pursuer is modeled as

a satellite with lateral thrusting capability using two-body orbital dynamics. The evader is modeled as an Intercontinental Ballistic Missile (ICBM) in its final boost phase, prior to burnout. The relative trajectory is propagated numerically to predicted impact time and then approximated by splines, eliminating the need to repeatedly propagate new trajectories when present conditions are varied. A search is then made for a new impact time and point that will minimize present interceptor velocity changes and final miss distance.

Six different variations of the general scheme are derived. The first scheme, presented in Chapter VI, uses a variable weighting factor and the principle of certainty equivalence to reduce velocity changes at the expense of final accuracy. The second scheme, also presented in Chapter VI, is a specialized version of the first, determining the velocity changes for zero miss. The third guidance algorithm in Chapter VI ignores the effects of gravity on the relative trajectory while attempting a zero miss solution. The fourth scheme, presented in Chapter VII, optimizes thrust spacing for a zero miss solution. The fifth scheme employs dual control techniques to reduce estimation error and is also presented in Chapter VII. The last algorithm is a new control approach that constrains the predicted miss distance to a function of final estimator error and is presented in Chapter VIII. Chapter IX summarizes the control strategies.

Target tracking is accomplished with a ranging device and line-of-sight sensors for in-plane and out-of-plane measurements. Noise corrupted data is processed through an eight state extended Kalman Filter with serial updates occurring every tenth of a second. The Kalman Filter equations are contained in Chapter X.

CHAPTER II

REVIEW OF LITERATURE

Much work has been done in the area of air-to-air intercept guidance. Guelman has derived a closed form solution for pure proportional navigation [1],[2] which is implemented in Chapter VI. Peturbation methods have been employed by Sridhar and Gupta [3]. Design procedures using optimal and stochastic control techniques abound [4]-[14] with variations of these techniques implemented in Chapters VI and VII. In the works cited above, the force of gravity is assumed to act equally on the pursuer and evader and is ignored in the relative dynamics. This 'flat earth' assumption is adequate for air-to-air encounters, but not for space-to-space. For orbital intercepts with large initial ranges the force of gravity will affect the relative trajectory and should be included in the equations of motion.

The literature for space-to-space guidance reveals many numerical approaches for determining present velocity for future rendezvous [15]-[21]. To date, analytic solutions for such intercepts exist only when the pursuer's impact conditions are pre-specified [19]. These works do not address hypervelocity intercept involving seconds, but are concerned with a much slower

rendezvous process involving hours or even days. Also, most of the literature reviewed assumed a passive, non-thrusting target. The literature that addressed thrusting targets was concerned with evasive maneuvering or 'gaming', the most recent being the paper by Menon and Calise [21]. A Defense Technology Information Center literature search revealed that the few papers addressing this problem are classified and therefore unavailable to the public.

The guidance schemes presented here attempt to minimize lateral velocity changes by varying the impact conditions through the use of splines. Splines were used by Johnson [16] in presenting a possible Earth-Mars transfer guidance algorithm. Dickmanns and Wells have used third order polynomials for general trajectory optimization [22], as well as Hargraves and Paris [23]. The splines eliminate the need to repeatedly propagate new trajectories when conditions are varied, resulting in faster searches. This feature makes them attractive for a hypervelocity orbital intercept where a fast and reasonably accurate numerical search is needed. Spline approximations are presented in Chapter V.

CHAPTER III

SYSTEM MODELING

In this chapter, the equations of motion for the evader and pursuer are developed, along with the necessary coordinate transformation for pursuer thrusting. Atmospheric drag will not be considered in the dynamics because the interceptor is assumed exo-atmospheric. Also, due to the pursuer's lateral thrusting limitation, the longitudinal axis will be assumed parallel to the pursuer's initial velocity vector.

It is convenient to transform the present coordinate frame to align the x axis with the pursuer's initial velocity vector. This is done by first rotating about the y axis until the z component of velocity is eliminated,

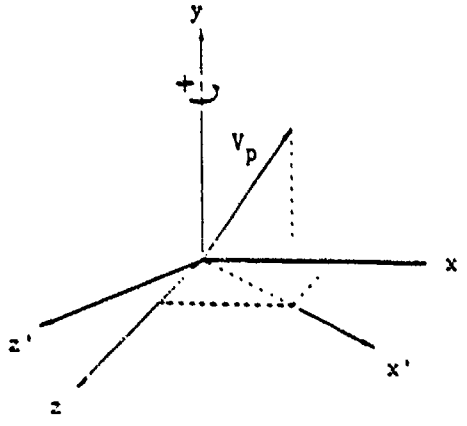

Figure 3-1. Rotation of coordinate frame about the y axis.

resulting in the following orthogonal transformation matrix:

$$[T_1] = \begin{bmatrix} \dfrac{\dot{x}_p}{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}} & 0 & \dfrac{\dot{z}_p}{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}} \\ 0 & 1 & 0 \\ \dfrac{-\dot{z}_p}{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}} & 0 & \dfrac{\dot{x}_p}{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}} \end{bmatrix} \qquad (3\text{-}1)$$

The second rotation is about the new z axis (z'), eliminating the y component of velocity.

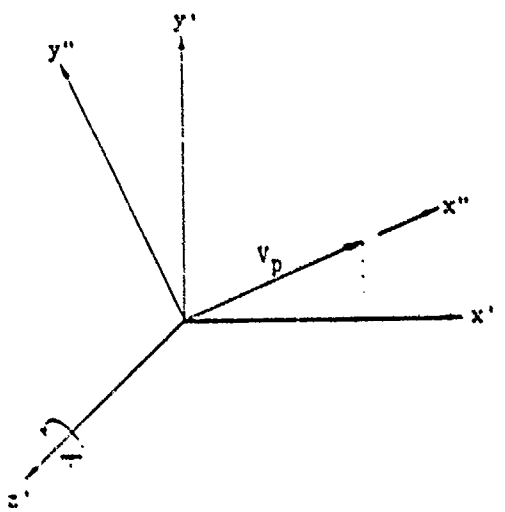

Figure 3-2. Rotation of the coordinate frame about the z axis.

This rotation yields the transformation matrix:

$$V_p = \sqrt{\dot{x}_p^2 + \dot{y}_p^2 + \dot{z}_p^2} \tag{3-2}$$

$$[T_2] = \begin{bmatrix} \frac{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}}{V_p} & \frac{\dot{y}_p}{V_p} & 0 \\ \frac{-\dot{y}_p}{V_p} & \frac{\sqrt{\dot{x}_p^2 + \dot{z}_p^2}}{V_p} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-3}$$

Multiplying the two matrices in the proper order produces the overall transformation matrix:

$$[T] = [T_2][T_1] \tag{3-4}$$

The pursuer is modeled as a satellite traveling in excess of twelve kilometers per second with lateral thrusting capability using two-body orbital dynamics. Thrusting is prohibited along the longitudinal (x) axis to prevent sensor contamination and to satisfy the structural constraints of having forward sensors and a large aft booster to achieve hypervelocity speed. The equations of motion are:

$$\ddot{x}_p = \frac{-\mu\, x_p}{(x_p^2 + y_p^2 + z_p^2)^{3/2}} \qquad (3\text{-}5)$$

$$\ddot{y}_p = \frac{-\mu\, y_p}{(x_p^2 + y_p^2 + z_p^2)^{3/2}} + a_y \qquad (3\text{-}6)$$

$$\ddot{z}_p = \frac{-\mu\, z_p}{(x_p^2 + y_p^2 + z_p^2)^{3/2}} + a_z \qquad (3\text{-}7)$$

where a_y and a_z are the lateral thrust accelerations, μ is the earth's gravitational constant, and the double dots denote the second derivative with respect to time.

The evader is modeled as an Intercontinental Ballistic Missile (ICBM) in its final boost phase using two-body orbital dynamics. For tracking purposes the intercept must occur prior to burnout. Acceleration due to thrusting is computed in the direction of the booster's velocity vector. The equations of motion are:

$$A = \frac{A_0}{1 - \dot{m}_0 t} \qquad (3\text{-}8)$$

$$\ddot{x}_E = \frac{-\mu\, x_E}{(x_E^2 + y_E^2 + z_E^2)^{3/2}} + \frac{A\, \dot{x}_E}{(\dot{x}_E^2 + \dot{y}_E^2 + \dot{z}_E^2)^{1/2}} \qquad (3\text{-}9)$$

$$\ddot{y}_E = \frac{-\mu\, y_E}{(x_E^2 + y_E^2 + z_E^2)^{3/2}} + \frac{A\, \dot{y}_E}{(\dot{x}_E^2 + \dot{y}_E^2 + \dot{z}_E^2)^{1/2}} \qquad (3\text{-}10)$$

$$\ddot{z}_E = \frac{-\mu\, z_E}{(x_E^2 + y_E^2 + z_E^2)^{3/2}} + \frac{A\, \dot{z}_E}{(\dot{x}_E^2 + \dot{y}_E^2 + \dot{z}_E^2)^{1/2}} \qquad (3\text{-}11)$$

where A is the present acceleration, A_0 the initial acceration, $\dot{m}_0$ the initial mass flow rate divided by mass, and t the time since ignition. The single dot denotes the first derivative with respect to time.

CHAPTER IV

PROBLEM STATEMENT AND TRUTH MODEL

Time-to-go and pursuer velocity changes are the control parameters that must be varied to minimize miss distance and fuel expended (i.e. velocity changes). This can be done by establishing a time remaining until intercept (time-to-go), propagating the equations of motion forward, and determining the miss distance. An iterative process can then be used to find the pursuer velocity needed to bring the miss distance to zero. The difference between current velocity and that needed for intercept, known as velocity-to-go, must be minimized. To accomplish this, the time-to-go is varied and the above procedure repeated until a minimum velocity-to-go is found.

The computation of needed velocity is time consuming because the equations of motion are nonlinear and do not lend themselves to closed form solution. These equations must be propagated numerically to intercept time whenever the initial velocity is varied. The above method will serve as the basis (truth) model for this control problem using the numerical techniques found in Maron [24].

A Newton-Raphson method for solving nonlinear systems is

employed to determine the proper values of the control parameters.

Let

$$\bar{u} = \begin{bmatrix} \overline{\Delta V_y} \\ \overline{\Delta V_z} \\ \overline{t_{go}} \end{bmatrix} \tag{4-1}$$

be a solution of the nonlinear system

$$\begin{bmatrix} f_1(\bar{u}) \\ f_2(\bar{u}) \\ f_3(\bar{u}) \end{bmatrix} = \begin{bmatrix} x_E(\overline{t_{go}}) - x_p(\overline{t_{go}}) \\ y_E(\overline{t_{go}}) - y_p(\overline{t_{go}}) - \Delta V_y \overline{t_{go}} \\ z_E(\overline{t_{go}}) - z_p(\overline{t_{go}}) - \Delta V_z \overline{t_{go}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4-2}$$

where t_{go} is the time-to-go and the pursuer's velocity changes are ΔV_y and ΔV_z. The effect of small velocity changes in (4-2) can be considered linear because the pursuer is assumed to travel at hypervelocity, resulting in a near straight-line trajectory. Any error caused by this assumption will be accounted for in the succeeding iteration when the proposed velocity change is incorporated in the nonlinear dynamics.

The initial control values must be incrementally changed to satisfy (4-2). A linear approximation of the $\underline{f}$ vector for

changes in $\underline{u}$ will yield approximate increments of the control parameters.

The linearized system becomes

$$[J] \begin{bmatrix} d\Delta V_y \\ d\Delta V_z \\ dt_{go} \end{bmatrix} = - \begin{bmatrix} f_1(\underline{u}) \\ f_2(\underline{u}) \\ f_3(\underline{u}) \end{bmatrix} \tag{4-3}$$

where J is the Jacobian matrix of the $\underline{f}$ vector evaluated at $\underline{u}$:

$$[J] = \begin{bmatrix} \frac{\partial f_1(\underline{u})}{\partial \Delta V_y} & \frac{\partial f_1(\underline{u})}{\partial \Delta V_z} & \frac{\partial f_1(\underline{u})}{\partial t_{go}} \\ \frac{\partial f_2(\underline{u})}{\partial \Delta V_y} & \frac{\partial f_2(\underline{u})}{\partial \Delta V_z} & \frac{\partial f_2(\underline{u})}{\partial t_{go}} \\ \frac{\partial f_3(\underline{u})}{\partial \Delta V_y} & \frac{\partial f_3(\underline{u})}{\partial \Delta V_z} & \frac{\partial f_3(\underline{u})}{\partial t_{go}} \end{bmatrix} . \tag{4-4}$$

Computing the partial derivatives yields

$$[J] = \begin{bmatrix} 0 & 0 & \{\dot{x}_E(t_{go}) - \dot{x}_p(t_{go})\} \\ -t_{go} & 0 & \{\dot{y}_E(t_{go}) - \dot{y}_p(t_{go}) - \Delta V_y\} \\ 0 & -t_{go} & \{\dot{z}_E(t_{go}) - \dot{z}_p(t_{go}) - \Delta V_z\} \end{bmatrix} \quad (4\text{-}5)$$

To determine changes in the $\underline{u}$ vector, $\underline{f}$ is multiplied by the negative inverse of J

$$\begin{bmatrix} d\Delta V_y \\ d\Delta V_z \\ dt_{go} \end{bmatrix} = -[J]^{-1} \begin{bmatrix} f_1(\underline{u}) \\ f_2(\underline{u}) \\ f_3(\underline{u}) \end{bmatrix} \quad (4\text{-}6)$$

$$[J]^{-1} = \begin{bmatrix} \dfrac{\dot{y}_E(t_{go}) - \dot{y}_p(t_{go}) - \Delta V_y}{\{\dot{x}_E(t_{go}) - \dot{x}_p(t_{go})\}t_{go}} & -1/t_{go} & 0 \\ \dfrac{\dot{z}_E(t_{go}) - \dot{z}_p(t_{go}) - \Delta V_z}{\{\dot{x}_E(t_{go}) - \dot{x}_p(t_{go})\}t_{go}} & 0 & -1/t_{go} \\ \dfrac{1}{\{\dot{x}_E(t_{go}) - \dot{x}_p(t_{go})\}} & 0 & 0 \end{bmatrix} \quad (4\text{-}7)$$

To find the control parameters the following procedure should be used. First, establish a time-to-go with zero velocity changes, a good choice being the time-to-go that yields the point of closest approach. This time-to-go is determined by propagating the orbits forward until a minimum relative distance is reached. Because the evader is assumed to be in its final boost phase throughout the intercept, this time-to-go will be less than or equal to time until ICBM thrust termination. Second, propagate the dynamic equations (3-5 thru 3-11) forward to the intercept time and determine the $\underline{f}$ vector from (4-2). Changes to the control parameters are then obtained from (4-6). The velocity changes are applied to the pursuer's initial conditions and the procedure is repeated with the updated time-to-go until convergence occurs. The resulting control parameters will drive the miss distance to zero with minimum velocity changes. The difference between needed and present velocity are sufficient to determine the pursuer's thrust profile.

CHAPTER V

SPLINE APPROXIMATIONS

As discussed in Chapter IV, numerical propagation of the dynamic equations is very time consuming. It would be convenient to approximate the relative trajectory by a polynomial, eliminating the need for repeated propagation. Cubic splines lend themselves well to this application [16], [22], [23]. The current and final states can be used to generate cubic splines along each axis of the form

$$x(t) = At_{go}^3 + Bt_{go}^2 + Ct_{go} + D \qquad (5\text{-}1)$$

By setting the current time to zero, D and C become the current position and velocity respectively, with time-to-go being the intercept time. Changes in velocity will be reflected only in the C coefficient and the final state can be easily determined for any intercept time. With this formulation, the determination of the spline coefficients is relatively simple. The current state gives D and C with no computations:

$$D = x(o) \qquad (5\text{-}2)$$

$$C = \dot{x}(o) \tag{5-3}$$

The A and B coefficients can be computed using the final states and (5-1) as follows:

$$x(t_{go}) = At_{go}^3 + Bt_{go}^2 + Ct_{go} + D \tag{5-4}$$

$$\dot{x}(t_{go}) = 3At_{go}^2 + 2Bt_{go} + C \tag{5-5}$$

Because there are only two unknowns in the above two equations, algebraic manipulation yields:

$$A = \frac{2[x(o) - x(t_{go})]}{t_{go}^3} + \frac{[\dot{x}(o) + \dot{x}(t_{go})]}{t_{go}^2} \tag{5-6}$$

$$B = \frac{3[x(t_{go}) - x(o)]}{t_{go}^2} + \frac{[2\dot{x}(o) + \dot{x}(t_{go})]}{t_{go}} \tag{5-7}$$

Figures depicting distance errors associated with these approximations are provided in Appendix A.

There is an added versatility in using splines. Should the system model be changed, only the spline coefficients need be changed. The search algorithms based on the splines will remain the same, operating with the new coefficients. This is very beneficial as it is far simpler to recompute the coefficients than

to alter the algorithms.

To ensure accuracy, new spline coefficients are computed every cycle time. To accomplish this, the truth model is propagated forward to predicted impact time to obtain the needed final states. By using these updated final states every iteration, propagated roundoff error is eliminated in the spline coefficient computations.

CHAPTER VI

OPTIMAL CONTROL FORMULATION USING CERTAINTY EQUIVALENCE

Changes in pursuer lateral velocity will affect final position, velocity and time. These effects can be easily computed with the relative trajectory modeled by splines in the coordinate system discussed in Chapter II. The optimal control problem is to find the intercept time that minimizes changes in pursuer velocity while ensuring a hit. Techniques to solve such problems are addressed by Bryson and Ho [25] and summarized in the following paragraphs.

PLAN A

To solve this problem, a relative spline equation is formed for each axis and a cost function is established. The cost function (L) incorporates velocity changes and miss distance multiplied by a weighting factor (K) and is represented as

$$L = \frac{K(x_1^2 + x_2^2 + x_3^2)}{2} + \frac{(\Delta V_y^2 + \Delta V_z^2)}{2} \qquad (6\text{-}1)$$

where the final relative state vector is determined from the spline equations:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x(t_{go}) \\ y(t_{go}) \\ z(t_{go}) \end{bmatrix} = \begin{bmatrix} A_x t_{go}^3 + B_x t_{go}^2 + C_x t_{go} + D_x \\ A_y t_{go}^3 + B_y t_{go}^2 + (C_y - \Delta V_y) t_{go} + D_y \\ A_z t_{go}^3 + B_z t_{go}^2 + (C_z - \Delta V_z) t_{go} + D_z \end{bmatrix} \quad (6\text{-}2)$$

The cost function must now be minimized with respect to the control vector $\underline{u}$:

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} t_{go} \\ \Delta V_y \\ \Delta V_z \end{bmatrix} \quad (6\text{-}3)$$

As stated in Bryson and Ho [25], it should be possible to find a set of controls such that

$$\frac{\partial L}{\partial \underline{u}} = \underline{0} \quad (6\text{-}4)$$

Three equations arise from (6-4) with three unknowns, expressed here in vector form as

$$\underline{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_2 \end{bmatrix} = \begin{bmatrix} K(x_1\dot{x}_1 + x_2\dot{x}_2 + x_3\dot{x}_3) \\ \Delta V_y - Kx_2 t_{go} \\ \Delta V_z - Kx_3 t_{go} \end{bmatrix} = 0 \qquad (6\text{-}5)$$

with $\dot{\underline{x}}$ being

$$\dot{\underline{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 3A_x t_{go}^2 + 2B_x t_{go} + C_x \\ 3A_y t_{go}^2 + 2B_y t_{go} + C_y - \Delta V_y \\ 3A_z t_{go}^2 + 2B_z t_{go} + C_z - \Delta V_z \end{bmatrix} \qquad (6\text{-}6)$$

As in the truth model, a Newton-Raphson method from Maron [24] is used to solve (6-5). It is important to note that this formulation differs from the truth model in two areas. First, a weighting factor has been introduced that allows a trade-off between miss distance and velocity changes. Zero miss distance is associated with infinite K, while zero K produces no velocity change. Second, the splines eliminate the need for repeated trajectory propagation, significantly reducing control parameter search time.

The Jacobian for (6-5) is

$$[J] = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & 0 \\ J_{31} & 0 & J_{33} \end{bmatrix} \tag{6-7}$$

where

$$J_{11} = K(x_1[6A_x t_{go} + 2B_x] + x_2[6A_y t_{go} + 2B_y] + x_3[6A_z t_{go} + 2B_z] + \dot{x}_1^2 + \dot{x}_2^2 + \dot{x}_3^2) \tag{6-8}$$

$$J_{12} = J_{21} = -K(x_2 + \dot{x}_2 t_{go}) \tag{6-9}$$

$$J_{13} = J_{31} = -K(x_3 + \dot{x}_3 t_{go}) \tag{6-10}$$

$$J_{22} = J_{33} = 1 + K t_{go}^2 \quad . \tag{6-11}$$

Changes in the control vector are determined by

$$d\underline{u} = \begin{bmatrix} dt_{go} \\ d\Delta V_y \\ d\Delta V_z \end{bmatrix} = -[J]^{-1} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \tag{6-12}$$

$$[J]^{-1} = \left[\begin{array}{ccc} J_{22} & -J_{12} & -J_{13} \\ -J_{12} & (J_{11} - J_{13}^2/J_{22}) & (J_{12}J_{13}/J_{22}) \\ -J_{13} & (J_{12}J_{13}/J_{22}) & (J_{11} - J_{12}^2/J_{22}) \end{array} \right] \Big/ (J_{11}J_{22} - J_{13}^2 - J_{12}^2) \qquad (6\text{-}13)$$

To execute this procedure, initialize time-to-go (preferably to the point of closest approach) and determine the spline coefficients for this initial trajectory. Compute the $\underline{x}$, $\dot{\underline{x}}$, and $\underline{h}$ vectors, in that order. Update the $\underline{u}$ vector using (6-12) and test for convergence. If convergence is not achieved recompute the above vectors and test again.

PLAN B

The formulation in this plan uses splines to determine the control parameters for a zero miss solution. This is a specialized version of Plan A where the weighting factor is set to infinity ($K=\infty$). Because the only control for miss in the longitudinal direction is time-to-go, x_1 in (6-2) is set equal to zero,

$$x_1 = A_x t_{go}^3 + B_x t_{go}^2 + C_x t_{go} + D_x = 0 \qquad (6\text{-}14)$$

and t_{go} solved using numerical techniques. With time-to-go established, (6-2) is again used with x_2 and x_3 equal to zero, yielding equations for the velocity changes:

$$\Delta V_y = A_y t_{go}^2 + B_y t_{go} + C_y + D_y/t_{go} \qquad (6\text{-}15)$$

$$\Delta V_z = A_z t_{go}^2 + B_z t_{go} + C_z + D_z/t_{go} \qquad (6\text{-}16)$$

This plan is computationally less burdensome than Plan A because the complexity of the search is reduced.

PLAN C

Within a few seconds of intercept the acceleration due to gravity will be nearly identical for the pursuer and evader. Also, the booster, still thrusting in the final boost phase, will travel in a near straight line along its velocity vector. Ignoring gravity terms in the relative dynamics leads to a simpler and faster solution, reducing the guidance to proportional navigation [1]. The relative trajectories are expressed as

$$D_A = \frac{A_o}{\dot{m}_o^2} \sum_{i=2}^{\infty} \frac{(\dot{m}_o t_{go})^i}{i(i-1)} \qquad (6\text{-}17)$$

$$x(t_{go}) = \left\{[x_E(0) - x_p(0)] + [\dot{x}_E(0) - \dot{x}_p(0)]t_{go} + \frac{D_A\dot{x}_E(0)}{\sqrt{\dot{x}_E^2(0) + \dot{y}_E^2(0) + \dot{z}_E^2(0)}}\right\} \tag{6-18}$$

$$y(t_{go}) = \left\{[y_E(0) - y_p(0)] + [\dot{y}_E(0) - \dot{y}_p(0) - \Delta V_y]t_{go} + \frac{D_A\dot{y}_E(0)}{\sqrt{\dot{x}_E^2(0) + \dot{y}_E^2(0) + \dot{z}_E^2(0)}}\right\} \tag{6-19}$$

$$z(t_{go}) = \left\{[z_E(0) - z_p(0)] + [\dot{z}_E(0) - \dot{z}_p(0) - \Delta V_z]t_{go} + \frac{D_A\dot{z}_E(0)}{\sqrt{\dot{x}_E^2(0) + \dot{y}_E^2(0) + \dot{z}_E^2(0)}}\right\} , \tag{6-20}$$

where D_A is the distance associated with thruster acceleration in the direction of booster velocity. As in Plan B, time-to-go is computed for zero miss on the x axis using (6-18), and then the velocity changes can be found from (6-19) and (6-20).

CERTAINTY EQUIVALENCE

It should be noted that all these techniques use the principle of certainty equivalence, where expected values from a state estimator are substituted for random variables [26]. The pursuer's states are assumed known, but because the evader's states must be estimated, the resulting system is stochastic. Optimal control formulation is based on a system that is deterministic. In applying the certainty equivalence principle, the stochastic system is replaced by a deterministic one, using the expected values of the random variables from the estimator.

There is a drawback to this technique in the sense that imperfect knowledge of the present state produces needless thrusting. Any errors in the present state estimate cause errors in the predicted final state. This results in the computation of velocity changes based on the incorrect final state. Future iterations produce similar results requiring the pursuer to thrust excessively.

This excessive thrusting can be reduced using stochastic control techniques. Three formulations are examined in the followint chapters. The first determines the optimum spacing of corrective thrusts for Plan B. The second uses dual control methods based on predicted error knowledge, such as filter covariance. The third constrains the miss distance to a function of predicted error knowledge, at the expense of accuracy.

CHAPTER VII

STOCHASTIC CONTROL

OPTIMUM SPACING OF CORRECTIVE THRUSTS

Corrective thrusting in the presence of state estimate errors can be optimally spaced to reduce fuel [27]. A control effectiveness ratio (ρ) is established to determine the spacing betwee. thrusts. This ratio directly yields thrust times when control effectiveness is a linear function of time.

For this formulation, the number of corrective thrusts (N) must be chosen to minimize the sum of thrusts (S_N), which is total ΔV. The behavior of ΔV and miss distance as a function of ρ can be produced through digital computation and is done as part of the simulation to determine the best value of ρ for Plan B.

To enhance understanding this technique, assume the control effectiveness ratio is two (ρ=2). This implies that corrective thrusting should take place when the control has half ($1/\rho$) the effect of the previous corrective thrust. If control effectiveness is a near-linear function of time, as is the case for a hypervelocity vehicle, then it will be halved at about half the time to impact since the last thrust. Thrusting will take

place at the start of the intercept, at one-half time-to-go, one fourth time-to-go, one-eigth time-to-go and so on. With ρ=3 the optimum thrust timing always occurs at a third of the time-to-go since the last correction. When the spacing is less than the estimator's cycle time, impact is imminent and thrust is terminated.

DUAL CONTROL FORMULATION

Optimal control solutions require perfect knowledge of the states, but in reality the information provided to the controller is only an estimate. As stated by Aoki [28], a theory of control should take into account the 'imperfectness' of information. This explains the need to incorporate statistical decision theory in control formulation. A solution that uses imperfect information will be sub-optimal, but it is desirable for such a solution to have the intrinsic characteristics of optimality [5]. Recognition that the control affects not only the state but also its uncertainty leads to a form of stochastic control known as dual control. This method not only drives the system to some final state, but attempts to improve state uncertainty along the way. The result is often greater accuracy and/or reduced fuel consumption.

A dual control method for controlling stochastic nonlinear systems with free end-time was developed by Tse and

Bar-Shalom [5]. This method differs from the optimal control formulation presented in Chapter VI. Instead of minimizing the cost function L of (6-1), the expected value of the cost function (E{L}) is minimized. To accomplish this, the final states and their covariances must be computed. This can be done by running the Extended Kalman Filter forward to predicted intercept time, as suggested by Tse, Bar-Shalom and Meier [4].

The solution involves establishing an expected cost function consisting of miss distance and covariance of each axis, along with the control. The cost function L from (6-1) is repeated here for convenience:

$$L = K \frac{(x_f^2 + y_f^2 + z_f^2)}{2} + \frac{(\Delta V_y^2 + \Delta V_z^2)}{2} \tag{7-1}$$

Assuming the estimates of the filter are Gaussian, the expected value is:

$$E\{L\} = K\left[\frac{\sigma_{xf}^2 + \sigma_{yf}^2 + \sigma_{zf}^2}{2} + \frac{\hat{x}_f^2 + \hat{y}_f^2 + \hat{z}_f^2}{2}\right]$$

$$+ E\left\{\frac{\Delta V_y^2 + \Delta V_z^2}{2}\right\} \tag{7-2}$$

The expected cost of (7-2) is conditioned on the controls. Two cases must be examined: the cost associated with the certainty equivalence (CE) solution (Plan A of Chapter VI) and the cost of deviating from that solution to improve the estimate. In this manner, the approximate best cost-to-go includes both estimation and control performance.

The expected cost of the CE solution is easily computed by determining the controls from Plan A and then running the Extended Kalman Filter forward to predicted impact time assuming measurement updates. The final filter data is then inserted into (7-2) to find the expected cost.

Finding the expected cost of deviating from the CE solution is computationally burdensome. The thrust direction that yields the greatest estimate improvement must first be determined. Thrusting in this direction will cause the expected miss distance to grow due to departure from the nominal (CE) path. It is therefore necessary to determine a new nominal path based on the deviation and include the control energy required for this path in the deviation cost estimate. Failure to do so may result in large expected miss distances that erroneously inflate the cost associated with deviation, causing the CE control of Plan A to always be chosen.

CHAPTER VIII

CERTAINTY CONTROL

As stated earlier, if the estimate is near perfect then optimal control should be used. For a less accurate estimate, dual control attempts to improve the measurement certainty, and thus the estimate, by expending control energy. This has been shown to work well if the certainty is a function of the control parameters [5]. Because range is included as a measurement, lateral deviations should not noticeably improve the estimate. For this reason, dual control techniques are not expected to work better than certainty equivalence formulations.

If the controls associated with cost do not affect state estimate certainty, fuel may be conserved by using that certainty to reduce the controls. By linking the controls to the certainty of the estimate, a near perfect estimate would yield the optimal control, with reduced control resulting from a poor estimate. To accomplish this, the predicted final states are constrained by a function of their variances at the final time. This form of control will be called certainty control and is implemented by establishing the cost function

$$L = \frac{\Delta V_y^2 + \Delta V_z^2}{2} \tag{8-1}$$

subject to the constraint:

$$f = \frac{x_f^2 + y_f^2 + z_f^2 - K[\sigma_{xf}^2 + \sigma_{yf}^2 + \sigma_{zf}^2]}{2} \leq 0 \tag{8-2}$$

where K is a weighting factor. The final state estimates (x_f, y_f, z_f) and their deviations (σ_{xf}, σ_{yf}, σ_{zf}) are determined by running the filter forward to predicted impact time without updates and then representing their time history with splines:

$$x_s = A_x t_{go}^3 + B_x t_{go}^2 + C_x t_{go} + D_x \tag{8-3}$$

$$y_s = A_y t_{go}^3 + B_y t_{go}^2 + C_y t_{go} + D_y \tag{8-4}$$

$$z_s = A_z t_{go}^3 + B_z t_{go}^2 + C_z t_{go} + D_z \tag{8-5}$$

$$x_f = x_s \tag{8-6}$$

$$y_f = y_s - \Delta V_y t_{go} \tag{8-7}$$

$$z_f = z_s - \Delta V_z t_{go} \tag{8-8}$$

$$\sigma_{xf} = A_{\sigma x} t_{go}^3 + B_{\sigma x} t_{go}^2 + C_{\sigma x} t_{go} + D_{\sigma x} \qquad (8\text{-}9)$$

$$\sigma_{yf} = A_{\sigma y} t_{go}^3 + B_{\sigma y} t_{go}^2 + C_{\sigma y} t_{go} + D_{\sigma y} \qquad (8\text{-}10)$$

$$\sigma_{zf} = A_{\sigma z} t_{go}^3 + B_{\sigma z} t_{go}^2 + C_{\sigma z} t_{go} + D_{\sigma z} \qquad (8\text{-}11)$$

Conceptually, the constraint produces a deviation sphere about the predicted impact point. If the predicted miss is inside or touching the sphere, thrusting is not necessary. If the predicted miss is outside the sphere, mininum thrusting is determined to bring the miss to the surface of the sphere. As the estimates improve, the constraint tightens and the sphere shrinks. The spline representations allow this stochastic problem to be solved deterministically. The constraint is adjoined to the cost function to form the Hamiltonian [29]:

$$H = L + \lambda f \qquad (8\text{-}12)$$

The partials of H with respect to the controls must equal zero:

$$\frac{\partial H}{\partial \Delta V_y} = \Delta V_y - \lambda y_f t_{go} = 0 \qquad (8\text{-}13)$$

$$\frac{\partial H}{\partial \Delta V_z} = \Delta V_z - \lambda z_f t_{go} = 0 \qquad (8\text{-}14)$$

$$\frac{\partial H}{\partial t_{go}} = \lambda(x_f\dot{x}_f + y_f\dot{y}_f + z_f\dot{z}_f - K[\sigma_{xf}\dot{\sigma}_{xf} + \sigma_{yf}\dot{\sigma}_{yf} + \sigma_{zf}\dot{\sigma}_{zf}]) = 0 \quad (8\text{-}15)$$

with the dot term expansions computed in Appendix B.

Equations 8-2, 8-13, 8-14, and 8-15 constitute four equations with four unknowns, which can be reduced two equations and two unknowns using (8-7) and (8-8). Substituting (8-7) into (8-13) yields

$$\Delta V_y = \frac{\lambda y_s t_{go}}{1+\lambda t_{go}^2} \quad (8\text{-}16)$$

$$y_f = \frac{y_s}{1+\lambda t_{go}^2} \quad . \quad (8\text{-}17)$$

In a similar manner, substituting (8-8) into (8-14) yields

$$\Delta V_z = \frac{\lambda z_s t_{go}}{1+\lambda t_{go}^2} \quad (8\text{-}18)$$

$$z_f = \frac{z_s}{1+\lambda t_{go}^2} \quad . \quad (8\text{-}19)$$

Equations 8-2 and 8-15 can now be solved in terms of λ and t_{go}. Once known, ΔV_y and ΔV_z can be determined from (8-16) and (8-18). The parameters λ and t_{go} can be found by numerical techniques using the Jacobian:

$$[J] \begin{bmatrix} dt_{go} \\ d\lambda \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \end{bmatrix} \tag{8-20}$$

$$f_1 = x_f^2 + y_f^2 + z_f^2 - \frac{K[\sigma_{xf}^2 + \sigma_{yf}^2 + \sigma_{zf}^2]}{2} \tag{8-21}$$

$$f_2 = x_f\dot{x}_f + y_f\dot{y}_f + z_f\dot{z}_f - K[\sigma_{xf}\dot{\sigma}_{xf} + \sigma_{yf}\dot{\sigma}_{yf} + \sigma_{zf}\dot{\sigma}_{zf}] \tag{8-22}$$

with the elements of the Jacobian matrix computed in Appendix B.

Should the states be perfectly known, the σ terms will be zero. In this case, the equations for certainty control reduce to the optimal control formulation for Plan B. Should the estimate be poor, the σ terms will be large and the inequality constraint of (8-2) is met with very little (if any) change in velocity. This demonstrates the principle of certainty control, where the certainty of the estimate affects control energy expenditure.

CHAPTER IX

SUMMARY OF CONTROL STRATEGIES

In this chapter a brief summary of all the control strategies is presented. It is intended to give the reader a basis for quick comparison. The cost function of each algorithm is given, along with the requirements for computation.

Plan A is an optimal control, certainty equivalence formulation that minimizes the cost function:

$$L = \frac{K(x_f^2 + y_f^2 + z_f^2)}{2} + \frac{(\Delta V_y^2 + \Delta V_z^2)}{2} \quad .$$

This algorithm requires an estimate of the final relative states.

Plan B is a certainty equivalence formulation that minimizes the cost function:

$$L = \frac{(\Delta V_y^2 + \Delta V_z^2)}{2}$$

subject to the constraint

$$f = x_f^2 + y_f^2 + z_f^2 = 0 \quad .$$

This algorithm requires an estimate of the final relative states.

Plan C is a certainty equivalence formulation with the same cost function and requirements as Plan B. The difference between these strategies is that gravity is ignored in the dynamic equations used to estimate the final relative states.

The optimal spacing of corrective thrusts also uses the same cost function and requirements as Plan B. In this strategy, however, the pursuer is not permitted to thrust every cycle time. Thrust timing is controlled by selecting a control effectiveness ratio to minimize control energy expenditure.

Dual Control is a stochastic control formulation that attempts to improve the estimate, and thus accuracy, by minimizing the cost function:

$$E\{L\} = K\left[\frac{\sigma_{xf}^2 + \sigma_{yf}^2 + \sigma_{zf}^2}{2} + \frac{\hat{x}_f^2 + \hat{y}_f^2 + \hat{z}_f^2}{2}\right]$$

$$+ E\left\{\frac{\Delta V_y^2 + \Delta V_z^2}{2}\right\} .$$

This algorithm requires estimates of the final relative states, their filter variances, and the relationship between control and variance.

Certainty Control is a new stochastic control formulation that reduces the control based on the certainty of the estimate by minimizing the cost function

$$L = \frac{(\Delta V_y^2 + \Delta V_z^2)}{2}$$

subject to the constraint

$$\frac{\hat{x}_f^2 + \hat{y}_f^2 + \hat{z}_f^2 - K\left[\sigma_{xf}^2 + \sigma_{yf}^2 + \sigma_{zf}^2\right]}{2} \leq 0 .$$

This algorithm requires estimates of the final relative states and their filter variances.

CHAPTER X

EXTENDED KALMAN FILTERING

Optimal estimates of the pursuer and evader are needed for the search algorithms to converge properly. Due to the nature of the dynamics and sensors, the relative position and velocity must be estimated from sampled nonlinear measurements. The estimation problem for a nonlinear system having continuous dynamics and discrete-time measurements is addressed by Gelb [30]. The Extended Kalman Filter (EKF) was chosen over other estimation methods because the optimal estimate is determinate. That is, the dynamics and observations of the pursuer and evader can be well predicted in the presence of Gaussian noise.

A summary of the continuous-discrete EKF algorithm from Gelb [30] follows. The equations for the state dynamics and measurements, as well as the computations of the partial derivatives, can be found in Appendix C. The system model is a continuous model of the state dynamics with white Gaussian noise $\{\underline{w}(t)\}$ added.

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t),t) + \underline{w}(t) \qquad (10\text{-}1)$$

$$\underline{w}(t) \sim N(\underline{0},Q(t)) \tag{10-2}$$

where $\underline{w}$ is a Gaussian (normal) random vector with mean $\underline{0}$ and covariance matrix Q. The measurement model is discrete and corrupted by white Gaussian noise $\underline{v}_k$:

$$\underline{z}_k = \underline{h}_k(\underline{x}(t_k)) + \underline{v}_k \tag{10-3}$$

$$\underline{v}_k \simeq N(\underline{0},R_k) \quad . \tag{10-4}$$

In either case, the noise is assumed uncorrelated for all t(k).

The state estimate, denoted by a hat, is propagated from (10-1) with

$$\dot{\hat{\underline{x}}} = \underline{f}(\hat{\underline{x}}(t),t) \tag{10-5}$$

and the error covariance P(t) is propagated by

$$\dot{P}(t) = F(\hat{\underline{x}}(t),t)P(t) + P(t)F^T(\hat{\underline{x}}(t),t) + Q(t) \tag{10-6}$$

$$F(\hat{\underline{x}}(t),t) = \left.\frac{\partial \underline{f}(\underline{x}(t),t)}{\partial \underline{x}(t)}\right|_{\underline{x}(t) = \hat{\underline{x}}(t)} \quad . \tag{10-7}$$

The measurements determine the gain matrix K_k through the equations

$$H_k(\hat{x}_k(-)) = \left.\frac{\partial h_k(x(t_k))}{\partial x(t_k)}\right|_{x(t_k) = \hat{x}_k(-)} \tag{10-8}$$

$$K_k = P_k(-)H_k^T(\hat{x}_k(-))[H_k(\hat{x}_k(-))P_k(-)H_k^T(\hat{x}_k(-)) + R_k]^{-1} \tag{10-9}$$

where the (-) symbolizes prior to update and the (+) after update.

With the gain matrix computed, the state estimate and error covariance can be updated by the following equations:

$$\hat{x}_k(+) = \hat{x}_k(-) + K_k[z_k - h_k(\hat{x}_k(-))] \tag{10-10}$$

$$P_k(+) = [I - K_k H_k(\hat{x}_k(-))]P_k(-) \quad . \tag{10-11}$$

It is advantageous to process measurements one at a time. This method, called serial updating [31], eliminates the requirement to compute a matrix inverse, thereby reducing computer load and avoiding the computational problems associated with inverting an ill-conditioned matrix. Also, measurements may be skipped without reformulating the filter equations, allowing greater flexibility in examining various tracking schemes. The simultaneous measurement components of the vector z_k can be considered serially over a very short time span.

CHAPTER XI

COMPUTER SIMULATION

A menu-driven program that simulates all the algorithms was written in FORTRAN 77 and run on a VAX 8600. The code for this program can be found in Appendix D. Six cases are examined to determine the accuracy and efficiency of each algorithm. In all cases the propagation (w) and measurement (v) noise properties associated with the filter are:

$$w_{x,y,z}(t) \sim N(0,\ 2.21516 \times 10^{-18}\ \frac{m^2}{s})$$

$$w_{\dot{x},\dot{y},\dot{z}}(t) \sim N(0,\ 5.52049 \times 10^{-20}\ \frac{m^4}{s^3})$$

$$w_A(t) \sim N(0,\ 4.29831 \times 10^{-12}\ \frac{m^2}{s^5})$$

$$w_{\dot{m}}(t) \sim N(0,\ 2.493241 \times 10^{-7}\ \frac{1}{s^3})$$

$$v_{\theta,\gamma}(k) \sim N(0,\ 1.0 \times 10^{-8})$$

$$V_R(k) \sim N(0,\ 1.0 \times 10^{-8} \times R^2\ \ m^2)$$

where θ is the out-of-plane line-of-sight angle, γ the in-plane line-of-sight angle and R is range.

The startup variances are:

$$\sigma^2_{xx} = \sigma^2_{yy} = \sigma^2_{zz} = 100\ m^2$$

$$\sigma^2_{\dot{x}\dot{x}} = \sigma^2_{\dot{y}\dot{y}} = \sigma^2_{\dot{z}\dot{z}} = 10\ \frac{m^2}{s^2}$$

$$\sigma^2_{AA} = .1\ \frac{m^2}{s^4}$$

$$\sigma^2_{\dot{m}\dot{m}} = 2.493241 \times 10^{-6}\ \frac{1}{s^2}$$

The pursuer's initial conditions for all cases are

$$x_p = -359899.441 \quad m$$

$$\dot{x}_p = 11991.950 \quad \frac{m}{s}$$

$$y_p = 6727335.870 \quad m$$

$$\dot{y}_p = 158.764 \qquad \frac{m}{s}$$

$$z_p = 0.0 \qquad m$$

$$\dot{z}_p = 0.0 \qquad \frac{m}{s}$$

with a lateral acceleration range of 3-60 m/s^2 in each axis.

The booster's characteristics are modeled as

$$A_o = 3.15788 \qquad \frac{m}{s^2}$$

$$\dot{m}_o = .01579 \qquad \frac{1}{s}$$

with time-to-go equaling 30 seconds.

A time lag of one tenth second is used for all algorithms when computing velocity changes. It is unrealistic to assume the filter can process measurements, the controller determine thrust commands, and the thrusters respond to those commands all instantaneously. One cycle time is chosen to allow the velocity changes computed in the previous cycle to be implemented in the present cycle. The controller routines are built to take this lag into account. Also, thrusting is not permitted during the first three seconds of an intercept to account for target acquisition.

The following page shows the evader initial condition for six cases. Case I represents a head on, in-plane intercept. Case II represents a head on, 10° out-of-plane intercept. Case III represents a head on, 20° out-of-plane intercept. Case IV represents an in-plane tail chase. Case V represents a 10° out-of-plane tail chase. Case VI represents a 20° out-of-plane tail chase.

Table 11-1. Evader Initial Conditions

		CASE I	CASE II	CASE III	CASE IV	CASE V	CASE VI
x_E	(m)	205720.173	202600.41	193342.193	-205720.173	-202600.401	-193342.193
$\dot{x}_E$	(m/s)	-6799.072	-6695.912	-6390.106	6805.072	6701.912	6396.106
y_E	(m)	6638526.297	6639834.472	6639829.206	6638526.297	6639834.427	6639829.206
$\dot{y}_E$	(m/s)	3101.494	3058.185	3058.501	3101.494	3058.185	3058.501
z_E	(m)	0.0	-38683.931	-76213.728	0.0	-38683.931	-76213.728
$\dot{z}_E$	(m/s)	3.0	1282.071	2523.104	3.0	1282.71	2523.104

CHAPTER XII

RESULTS

A history of the miss and velocity changes with respect to multipliers is generated for those algorithms requiring multipliers (see Figures 12-1 through 12-24). A specific multiplier is then chosen for each plan for inclusion in the tables that follow.

One hundred Monte Carlo runs are generated per plan per case. The mean of each set of runs is adequate for judging relative performance. This performance is recorded in the six tables that follow the figures.

Appendix E contains the in-plane thrust profiles for Cases I and V of all plans. In the appendix, each profile uses the same random seed for startup to show the effect of estimate uncertainty on the various control strategies.

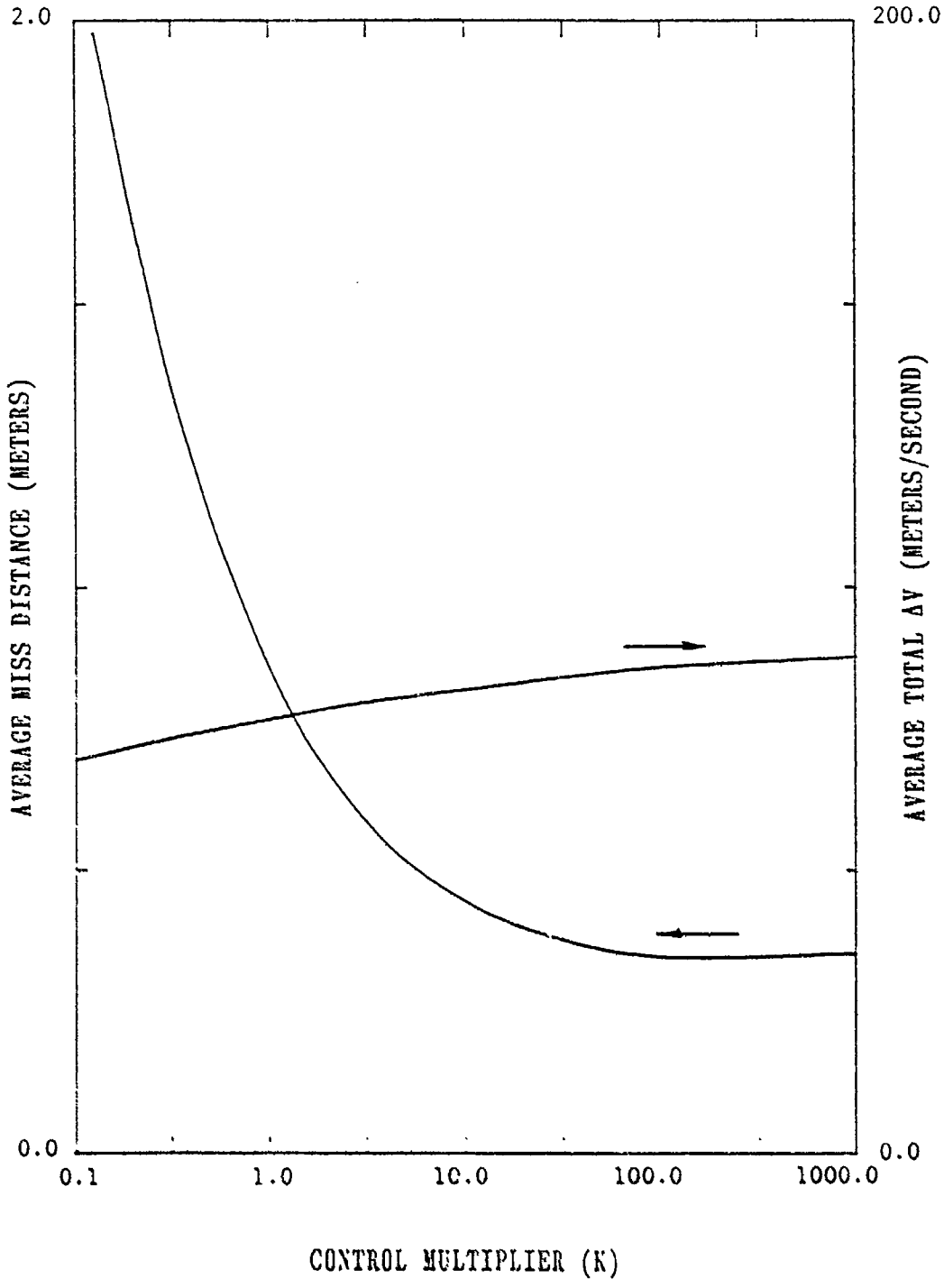

Figure 12-1. Performance of Plan A for Case I.

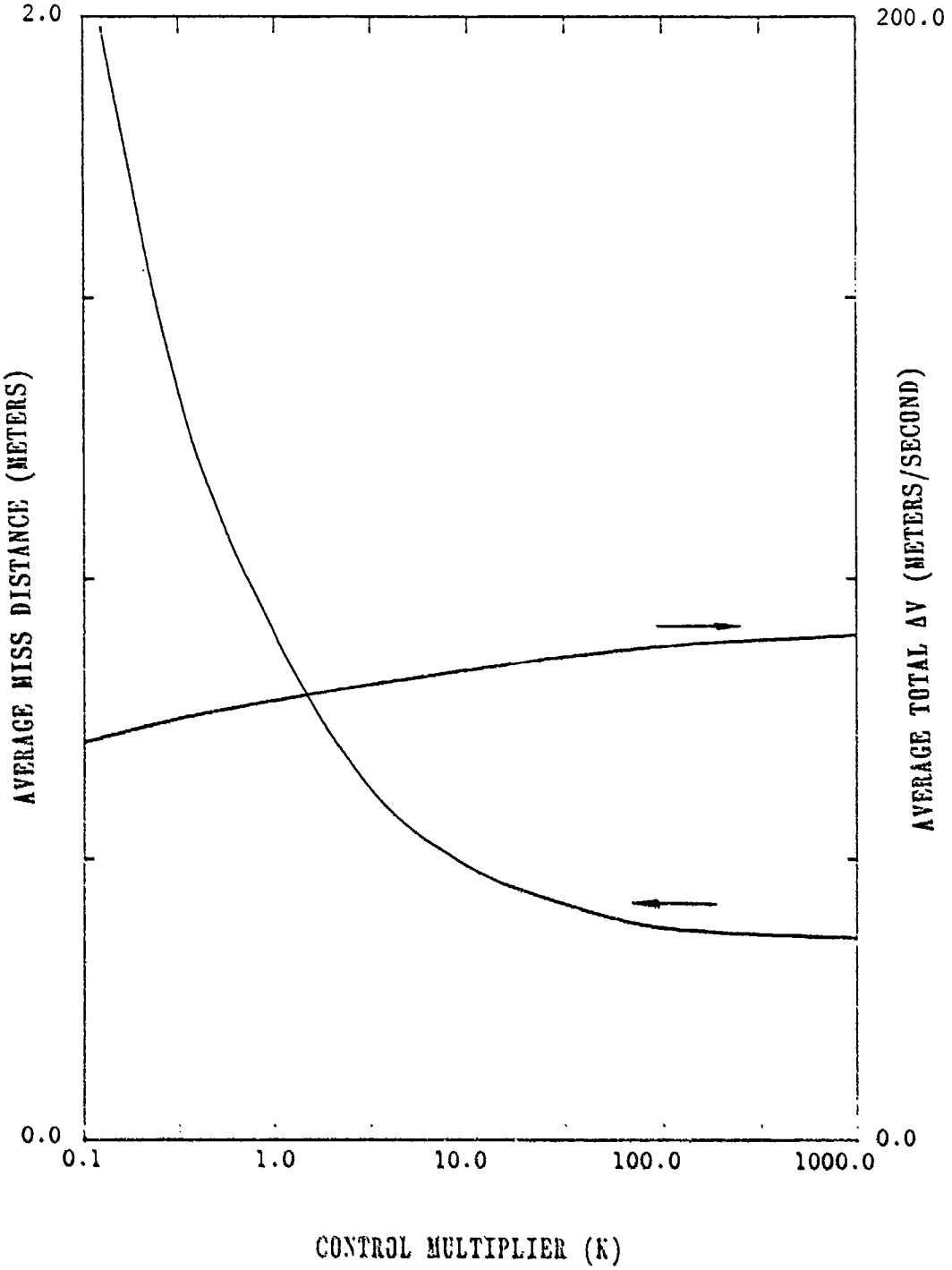

Figure 12-2. Performance of Plan A for Case II.

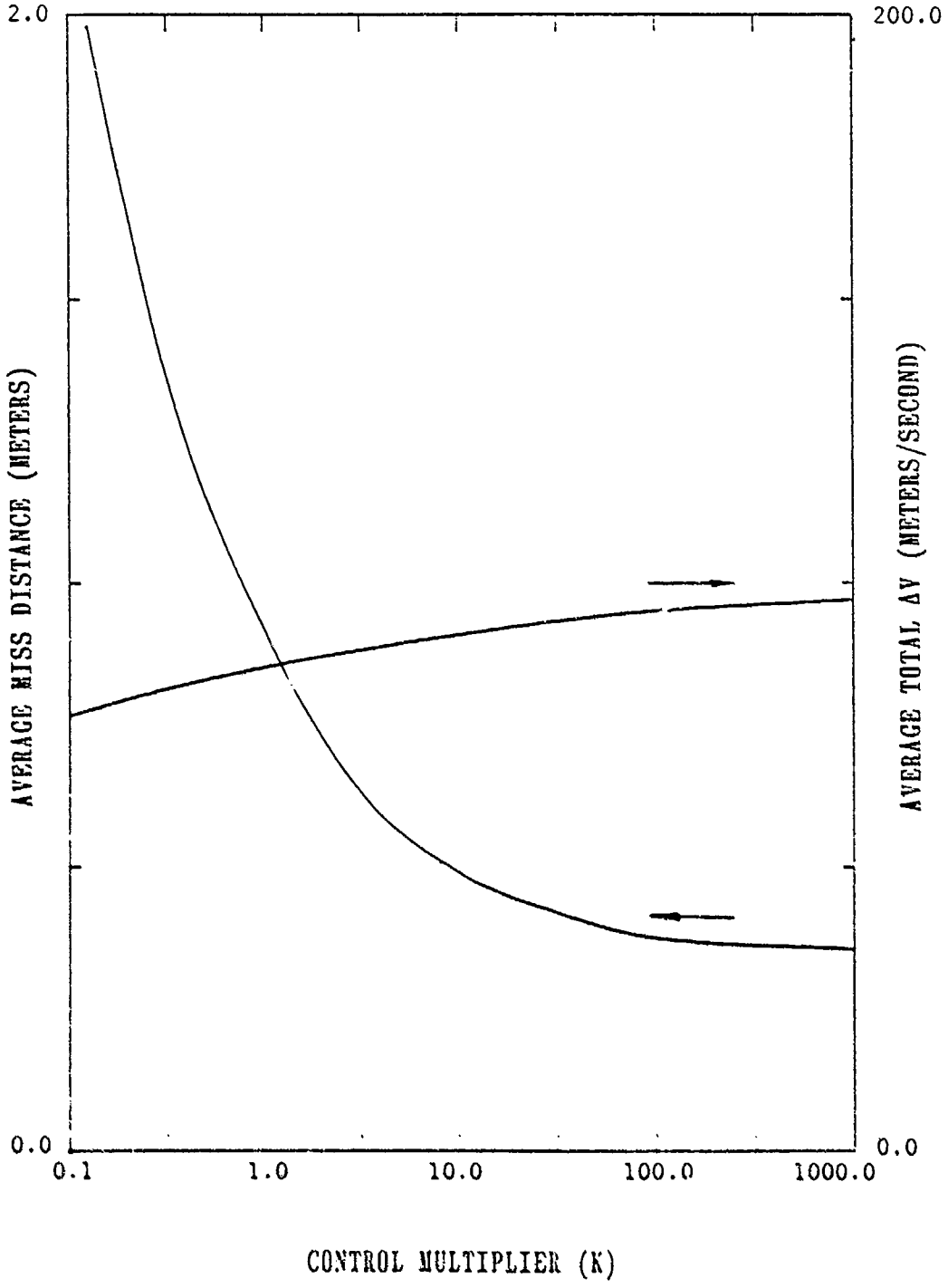

Figure 12-3. Performance of Plan A for Case III.

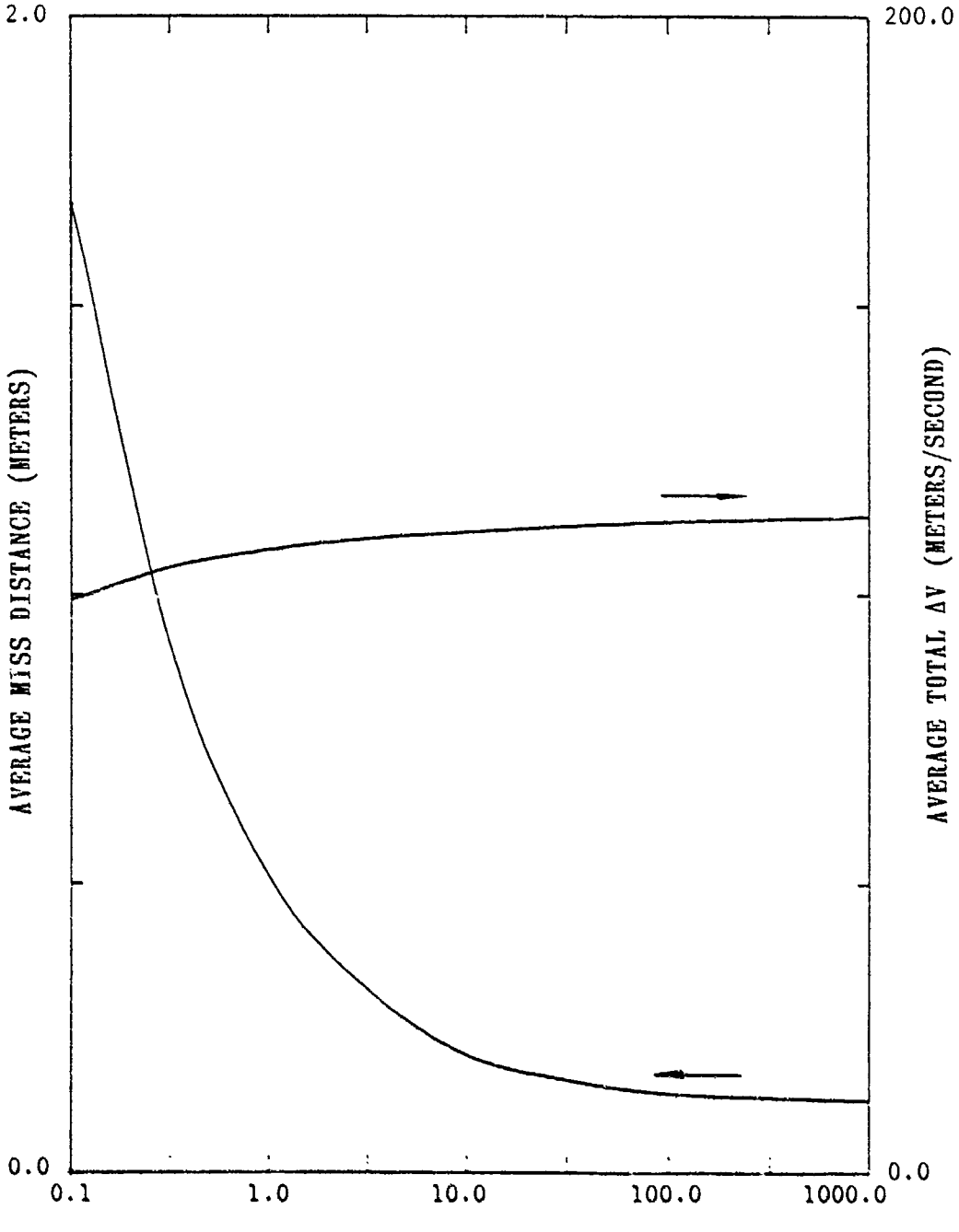

Figure 12-4. Performance of Plan A for Case IV.

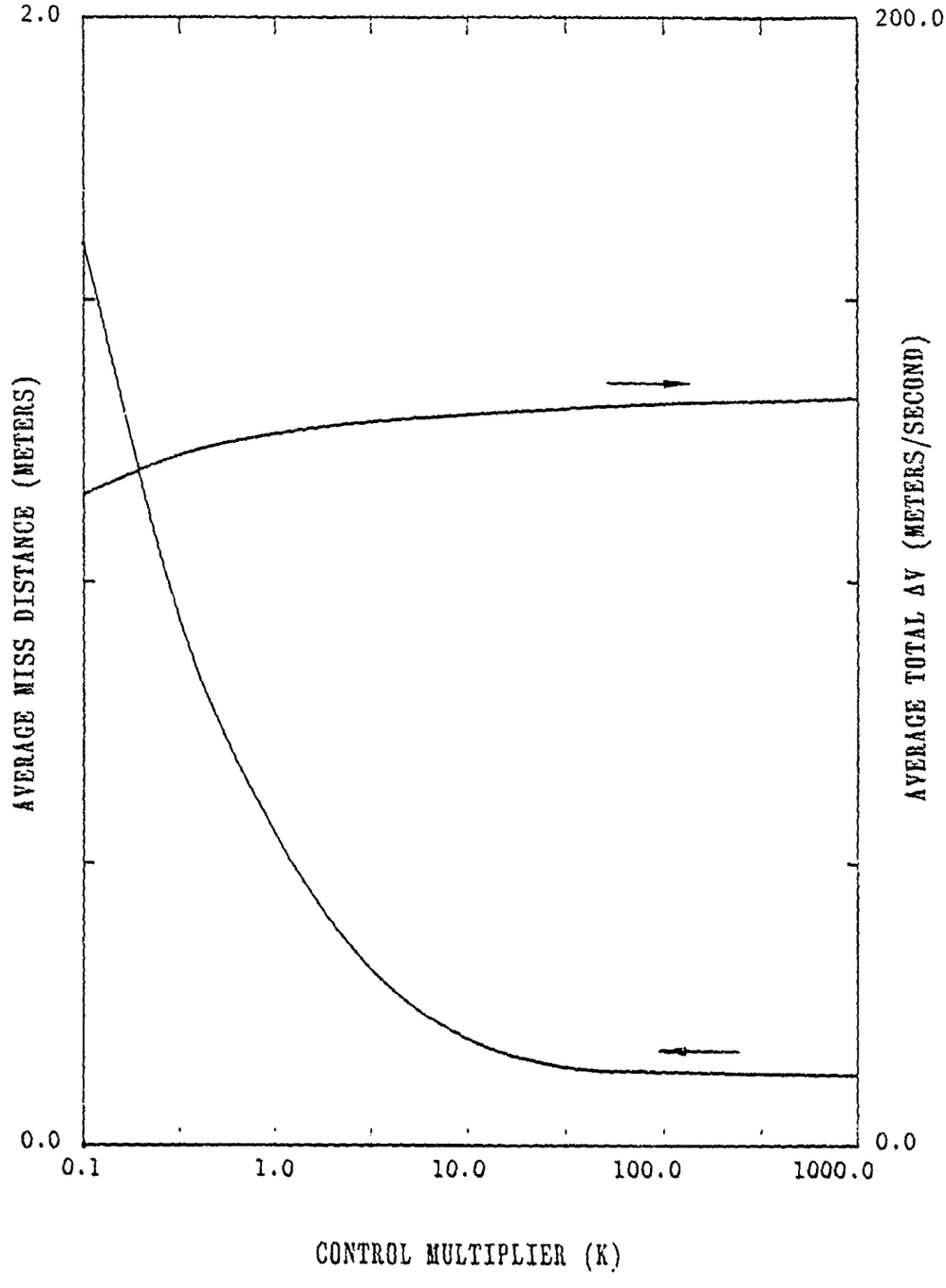

Figure 12-5. Performance of Plan A for Case V.

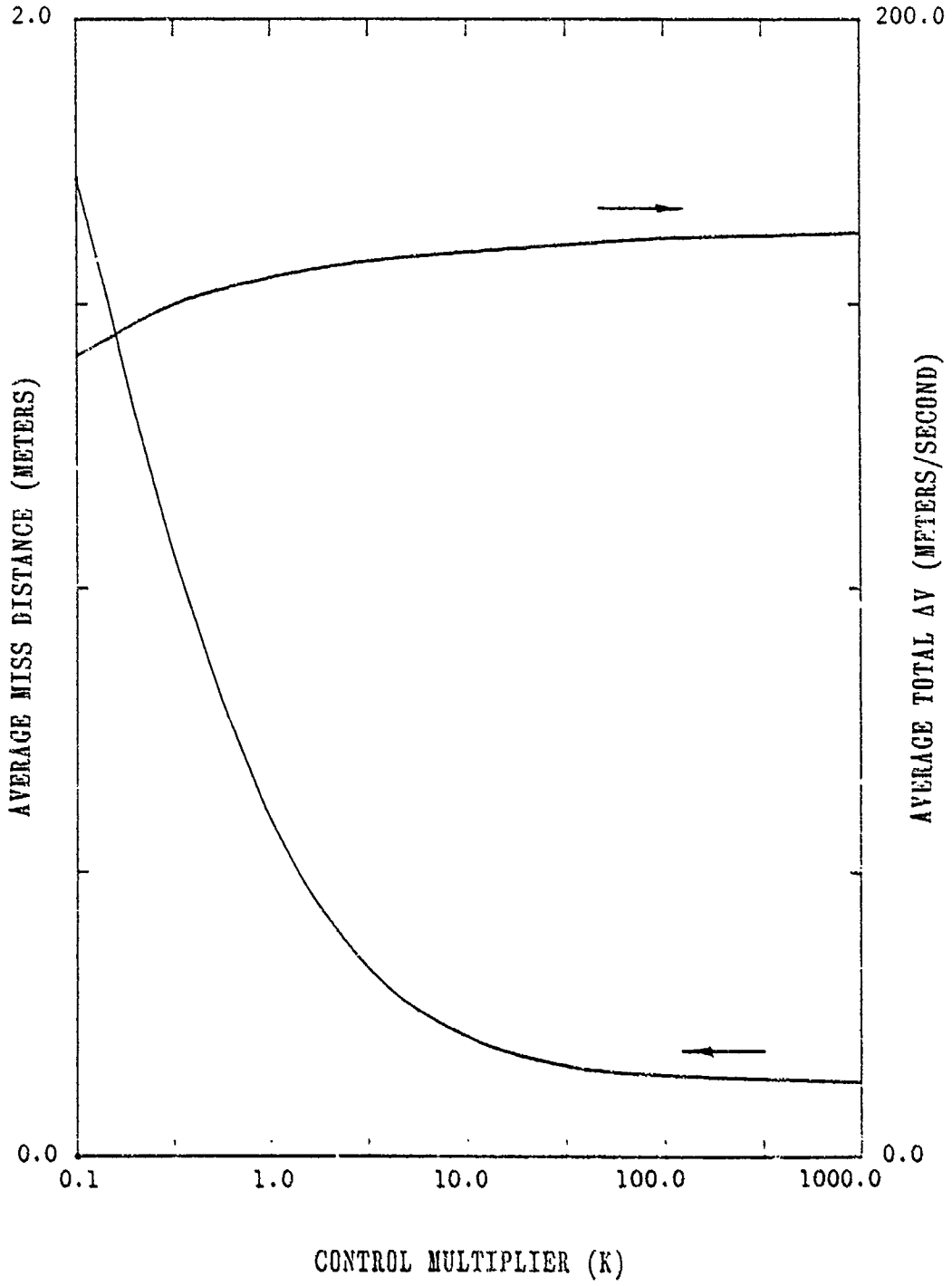

Figure 12-6. Performance of Plan A for Case VI.

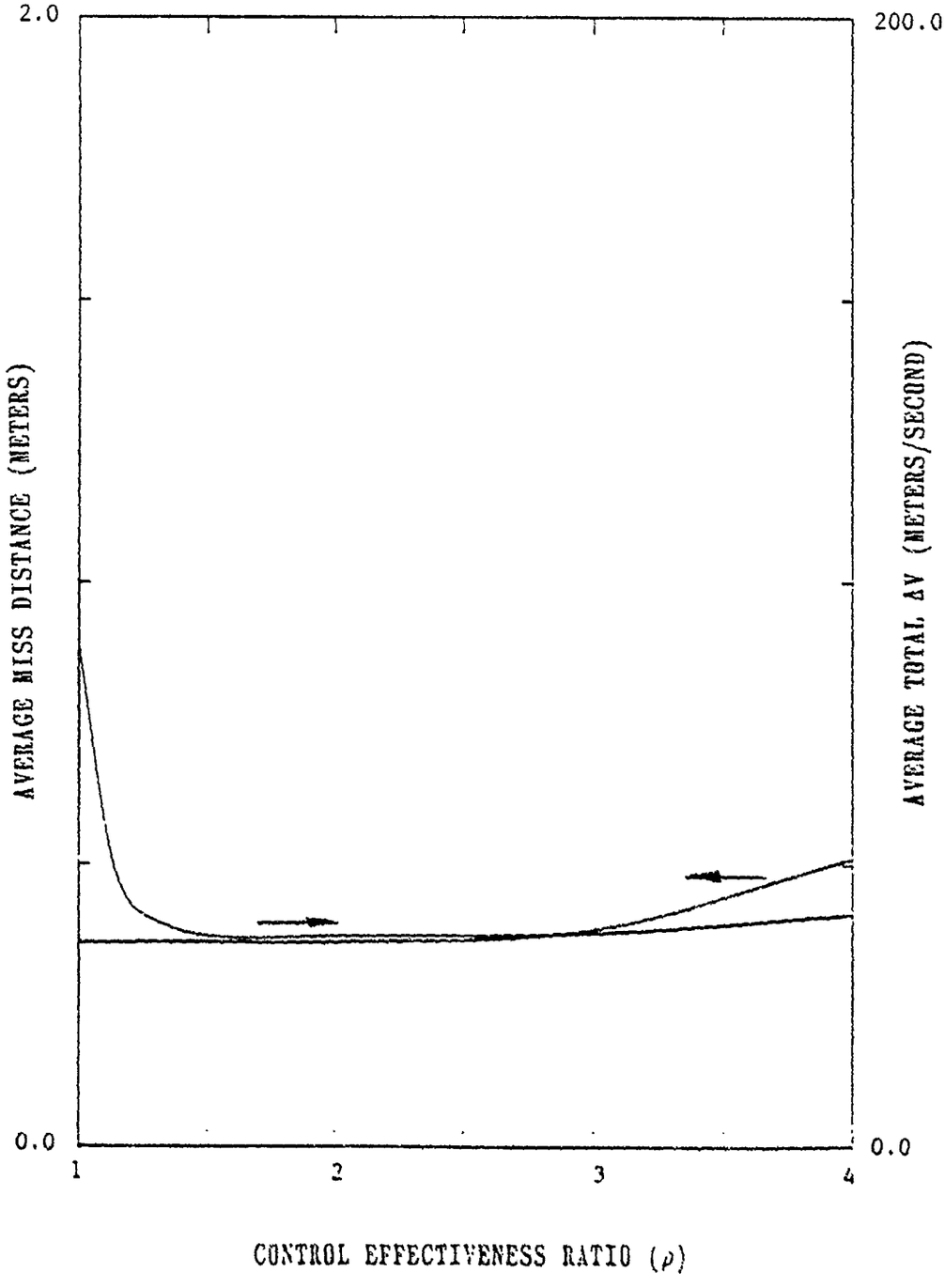

Figure 12-7. Performance of Optimum Thrust Spacing for Case I.

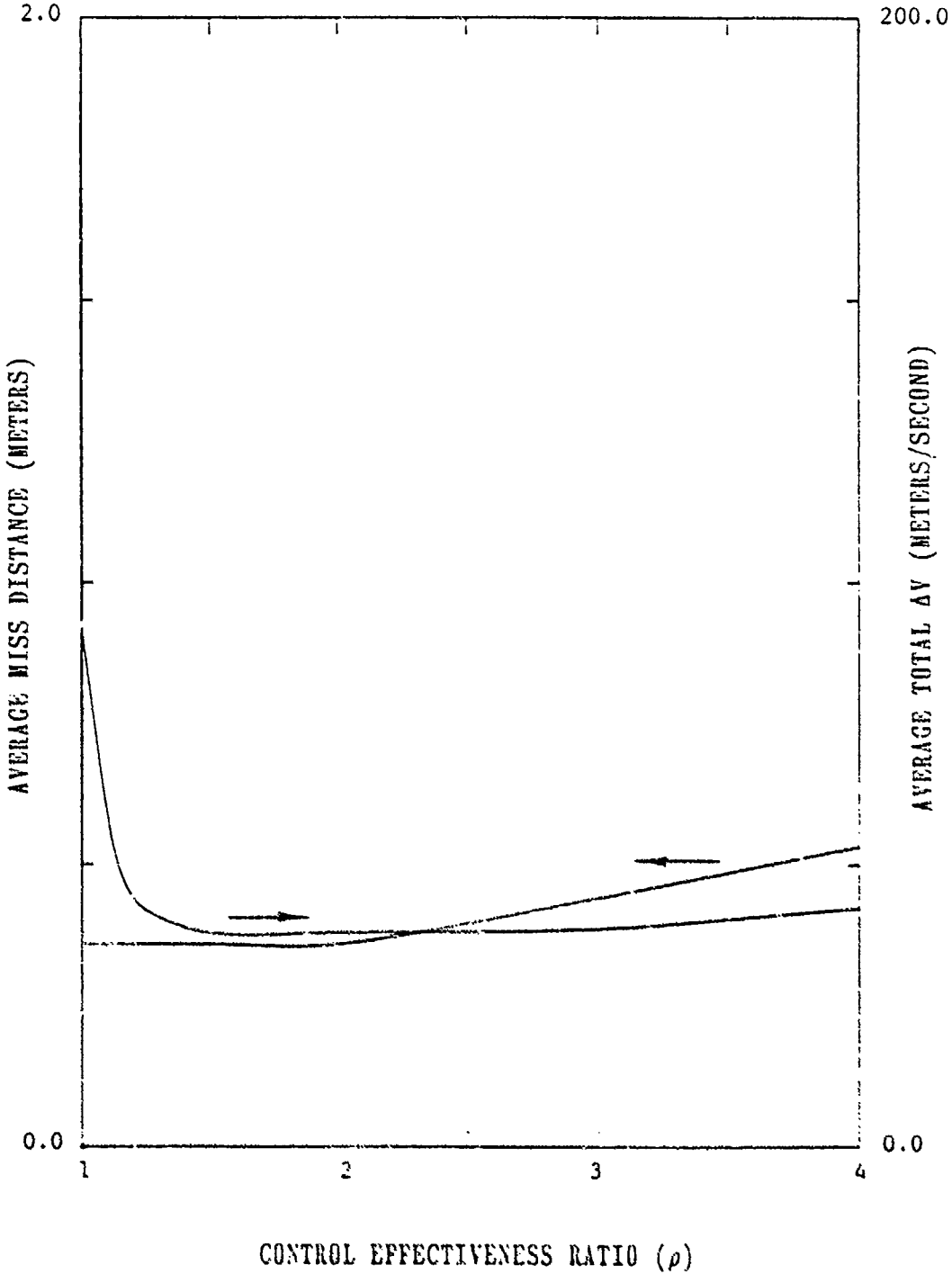

Figure 12-8. Performance of Optimum Thrust Spacing for Case II.

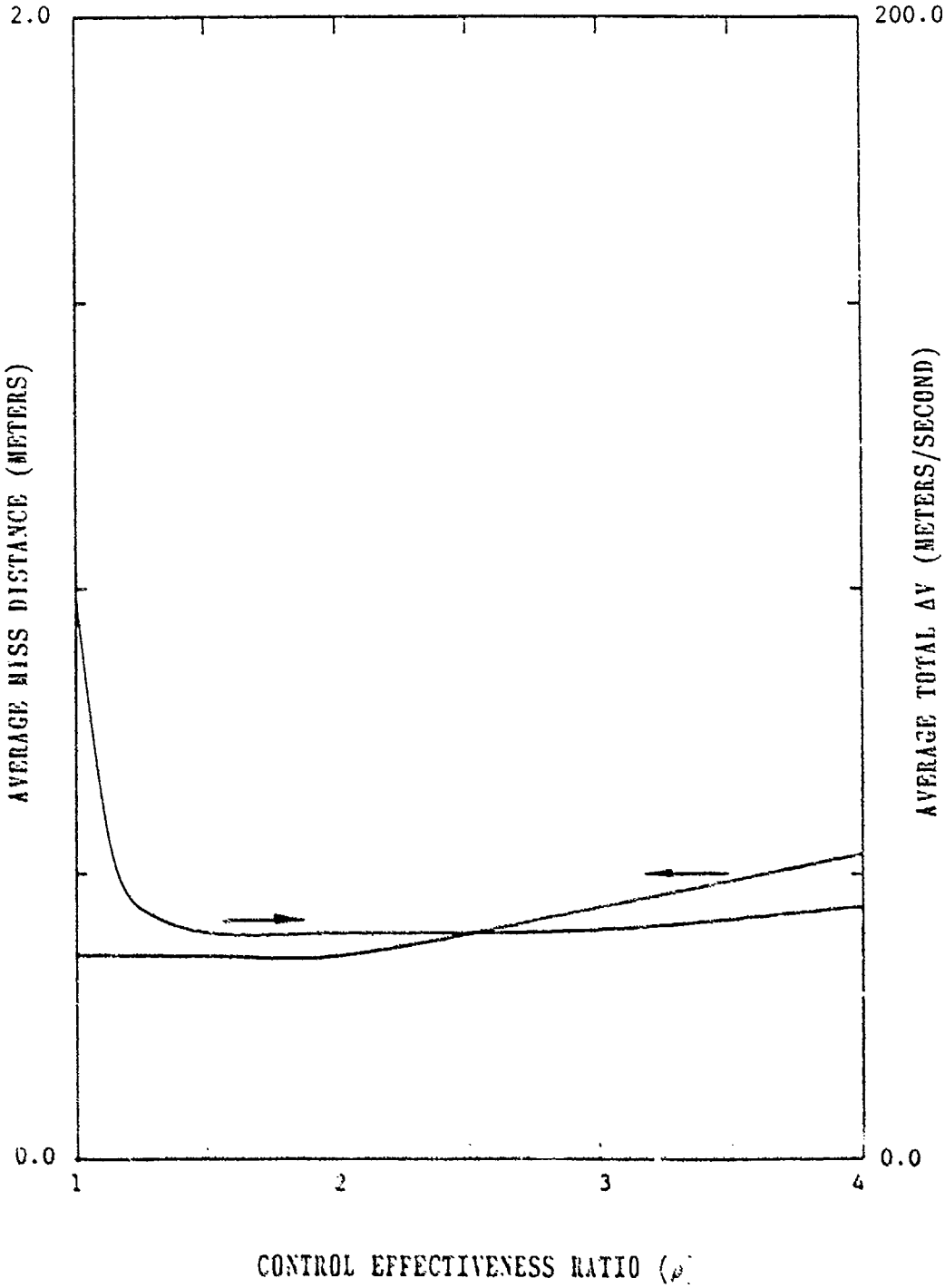

Figure 12-9. Performance of Optimum Thrust Spacing for Case III.

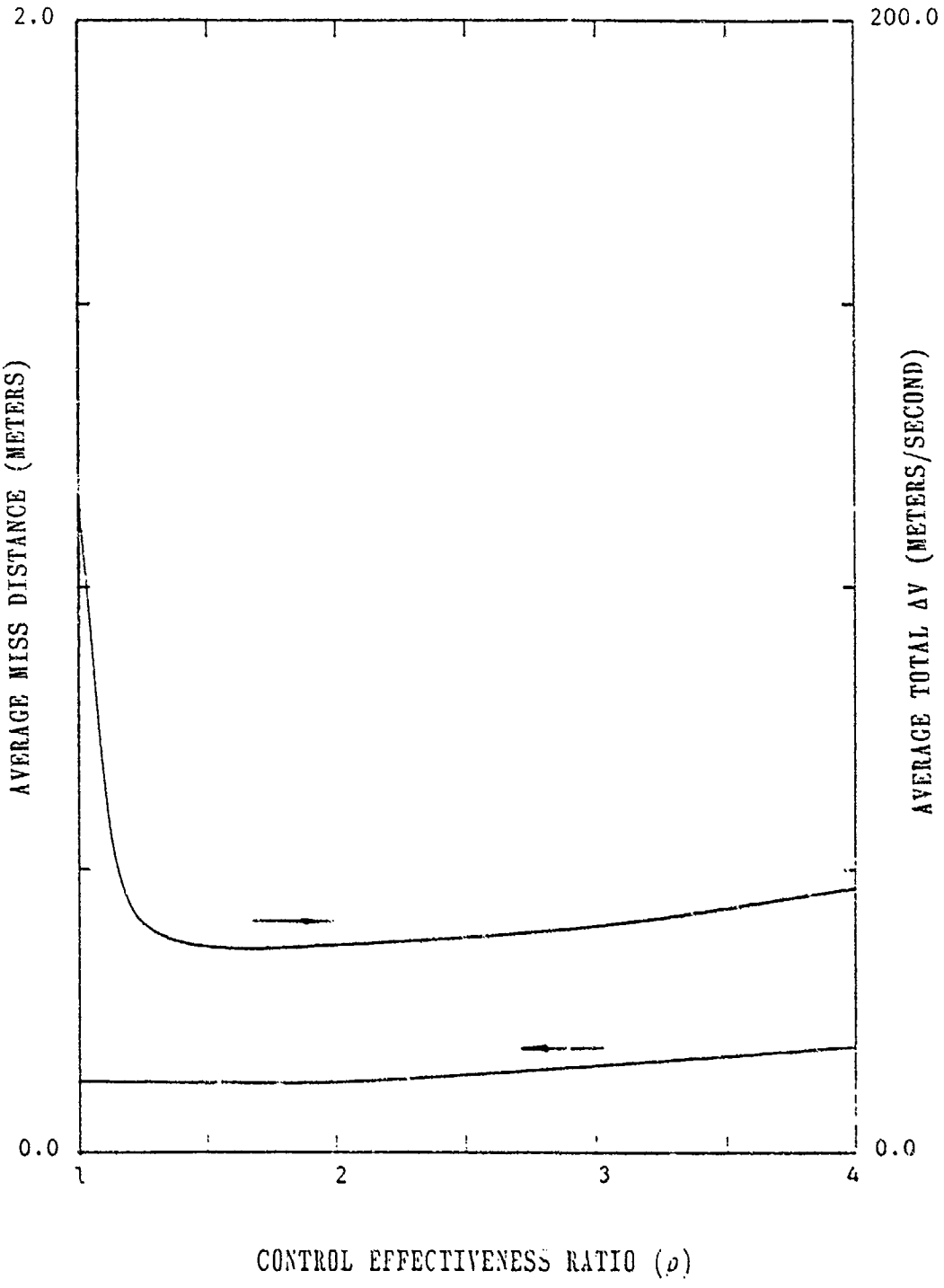

Figure 12-10. Performance of Optimum Thrust Spacing for Case IV.

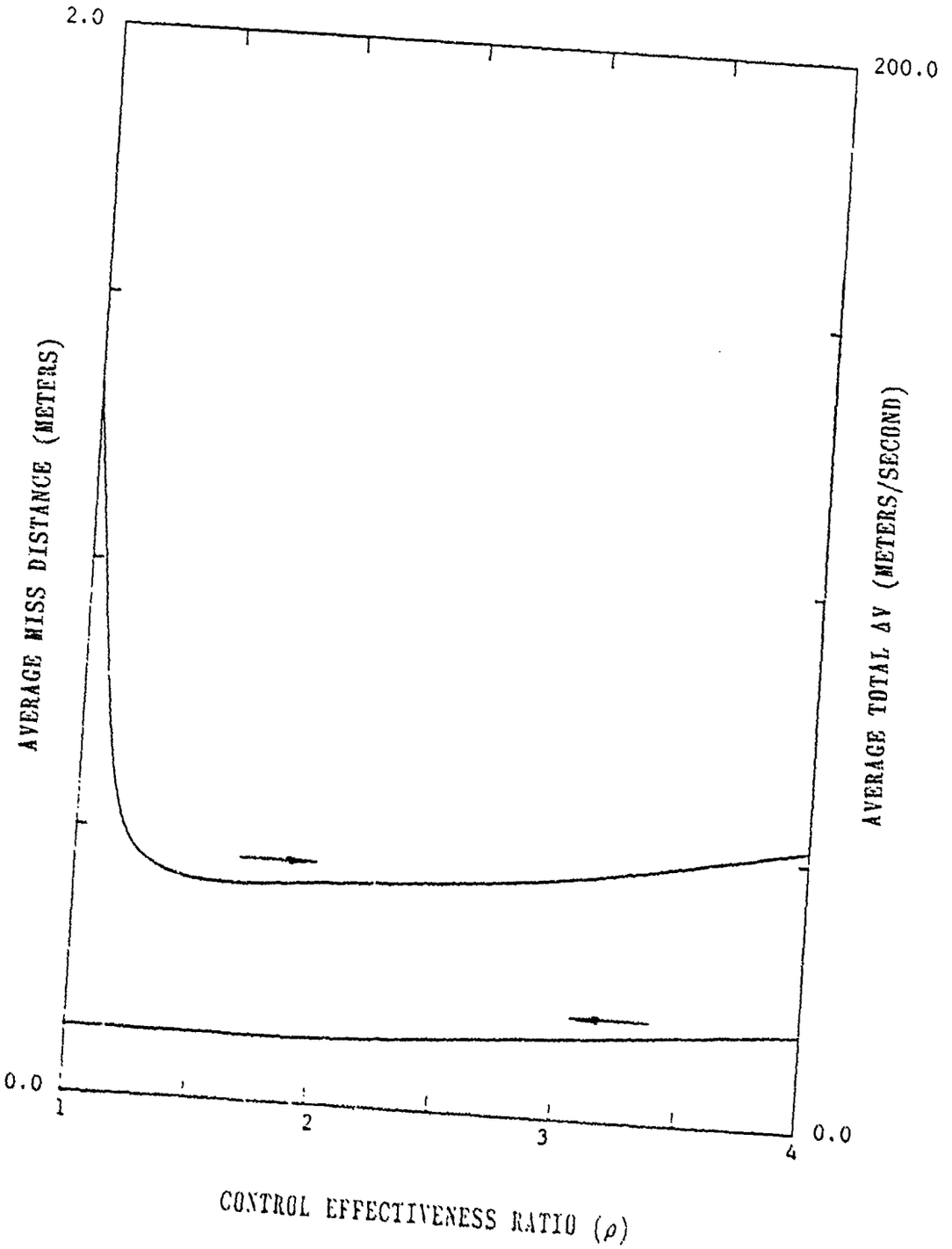

Figure 12-11. Performance of Optimum Thrust Spacing for Case V.

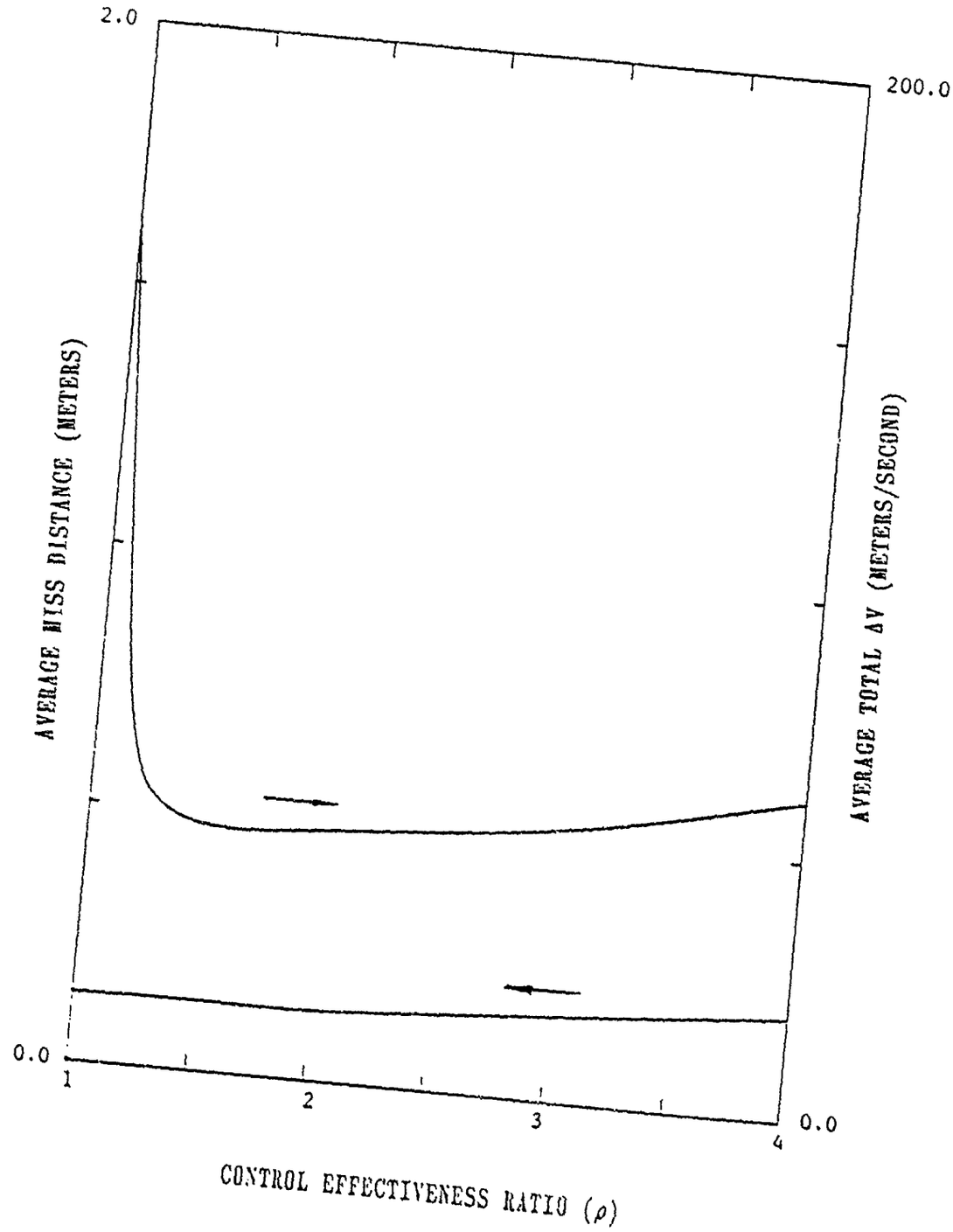

Figure 12-12. Performance of Optimum Thrust Spacing for Case VI.

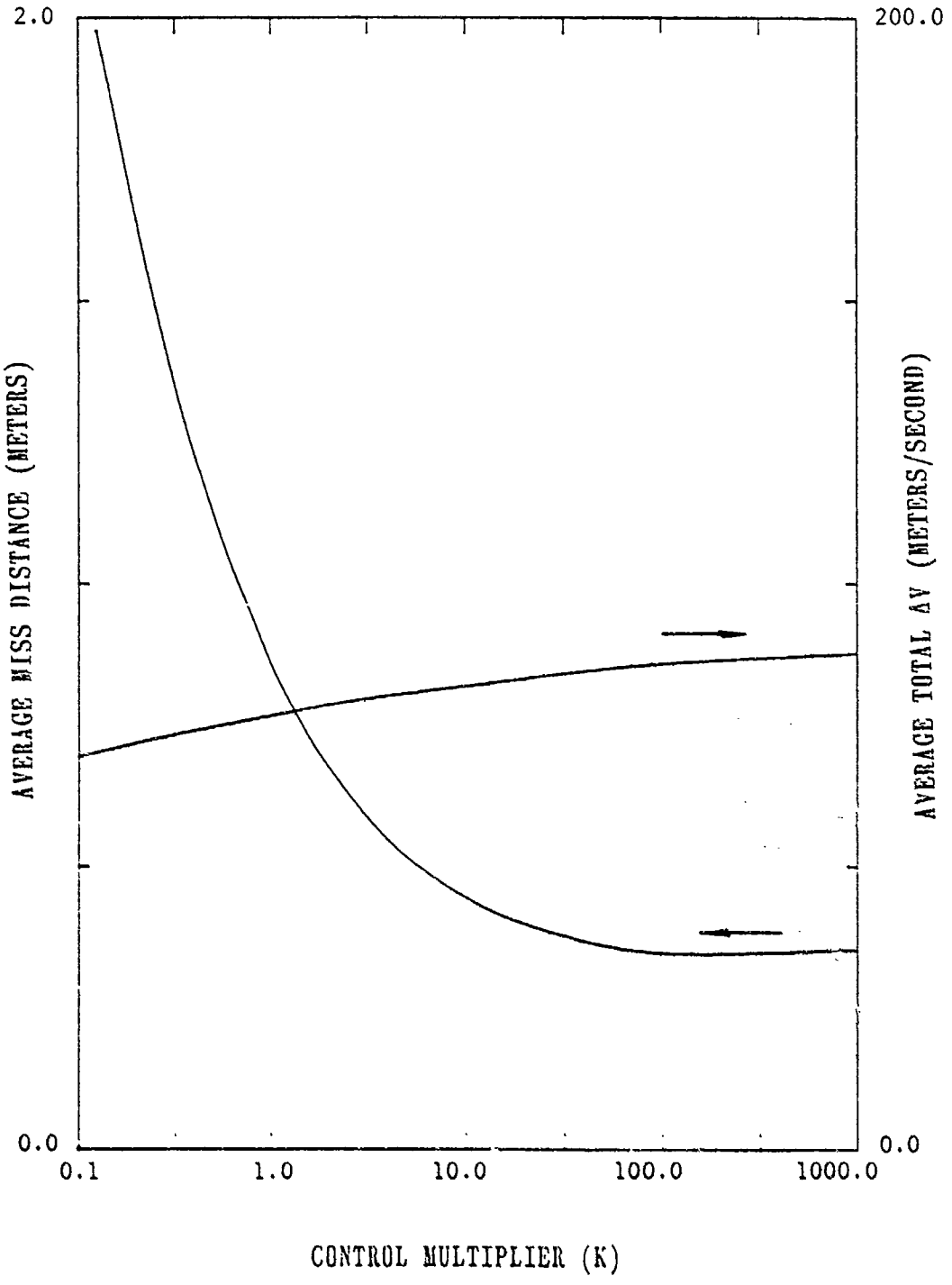

Figure 12-13. Performance of Dual Control for Case I.

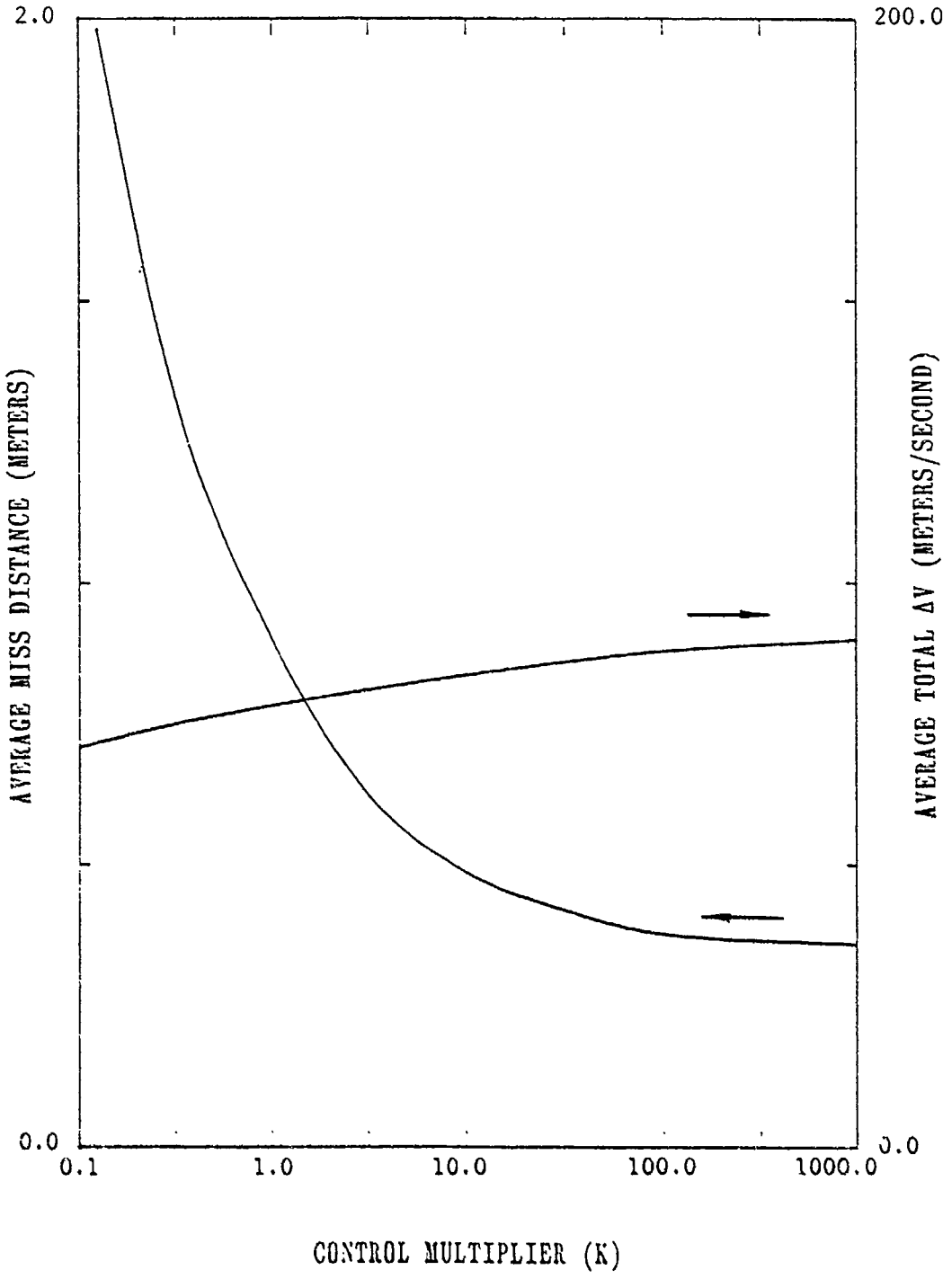

Figure 12-14. Performance of Dual Control for Case II.

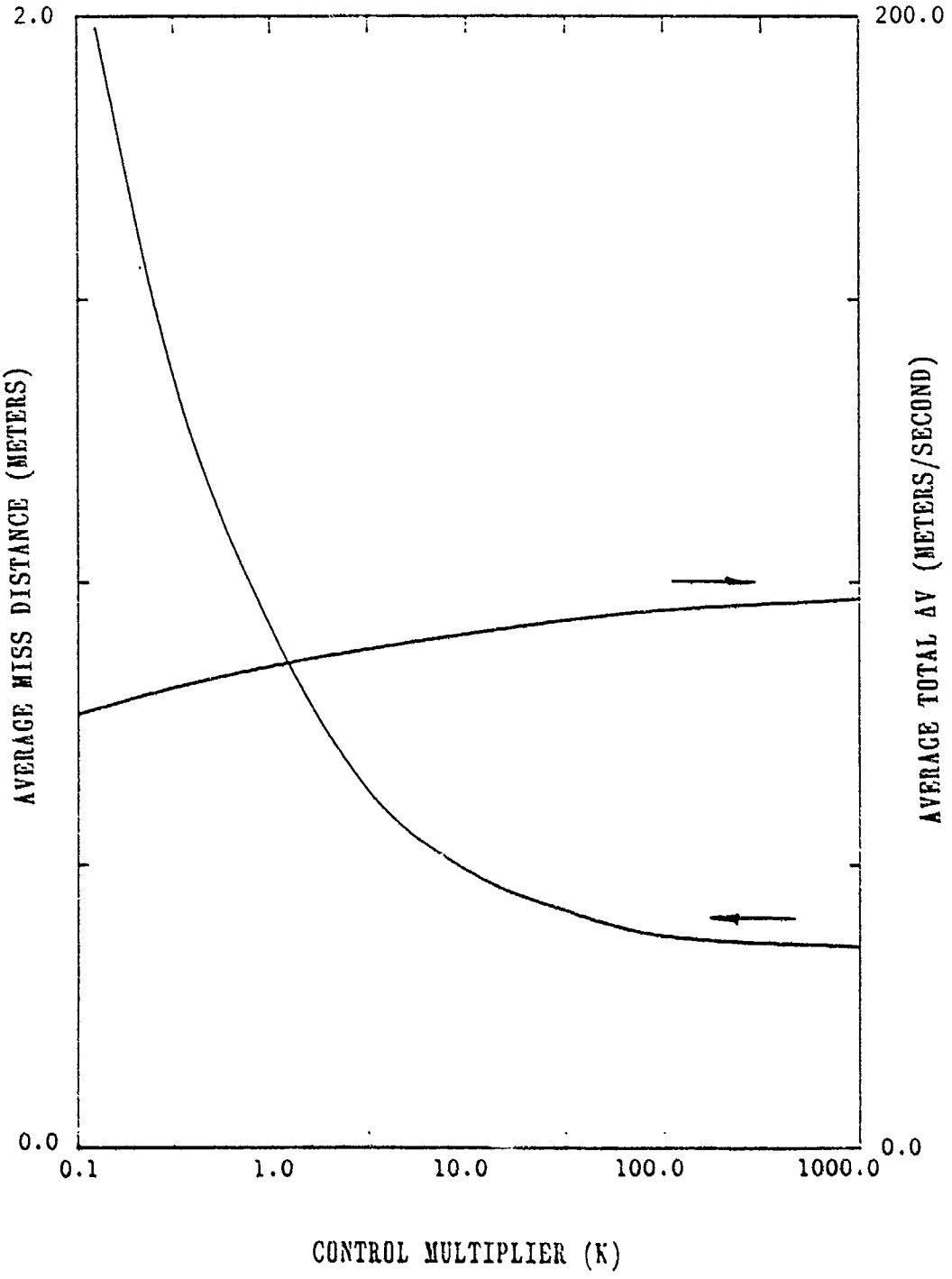

Figure 12-15. Performance of Dual Control for Case III.

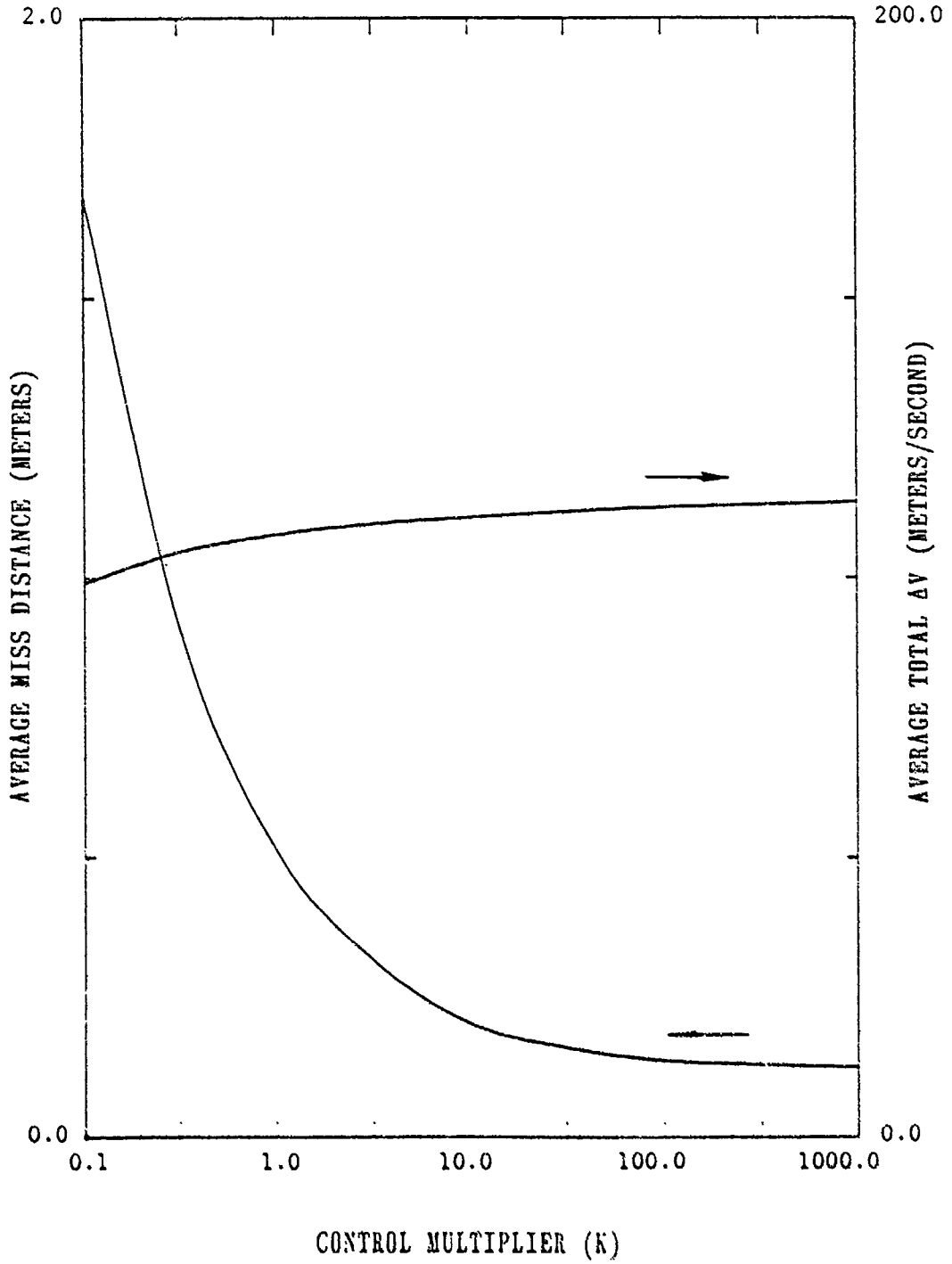

Figure 12-16. Performance of Dual Control for Case IV.

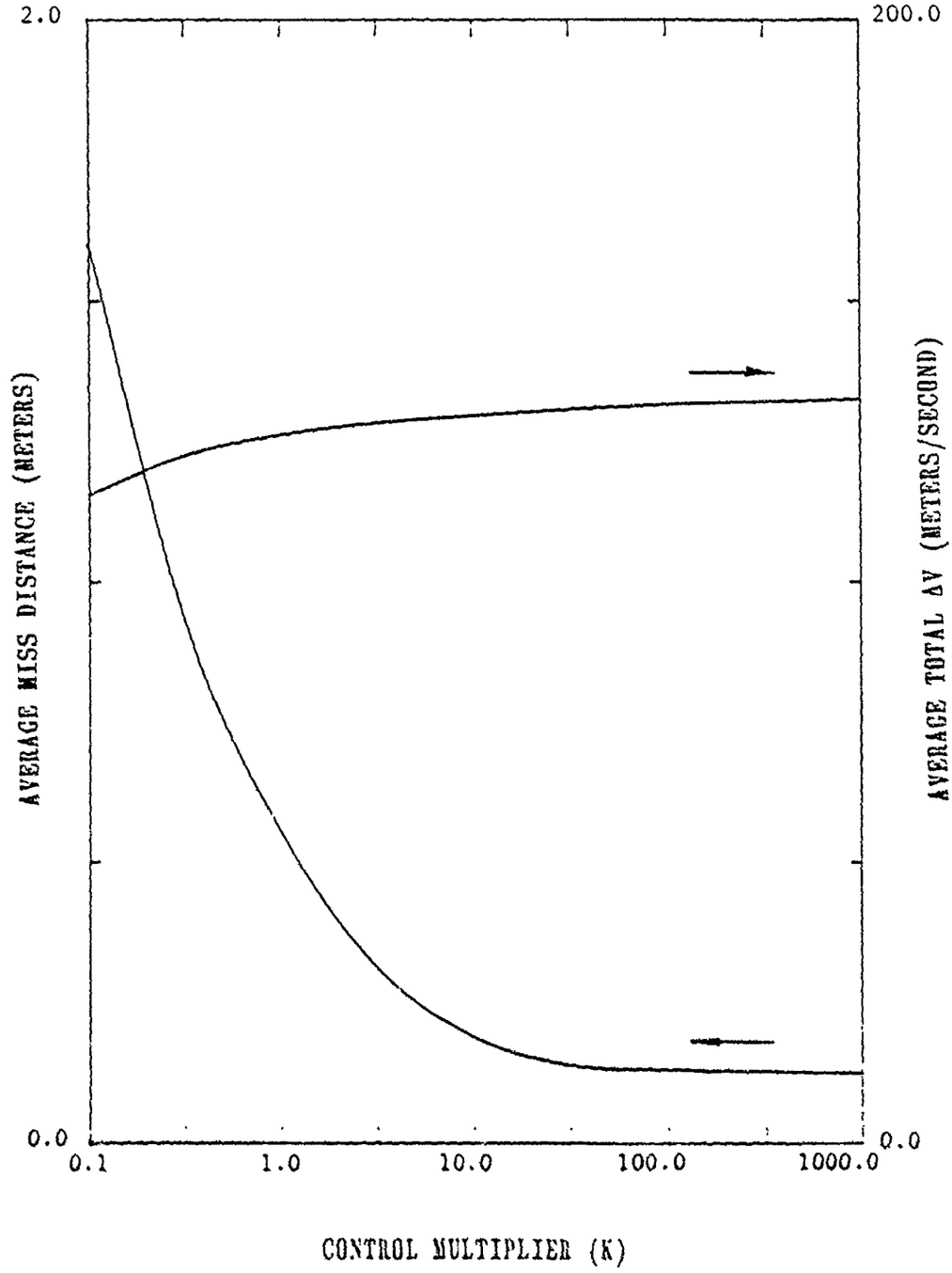

Figure 12-17. Performance of Dual Control for Case V.

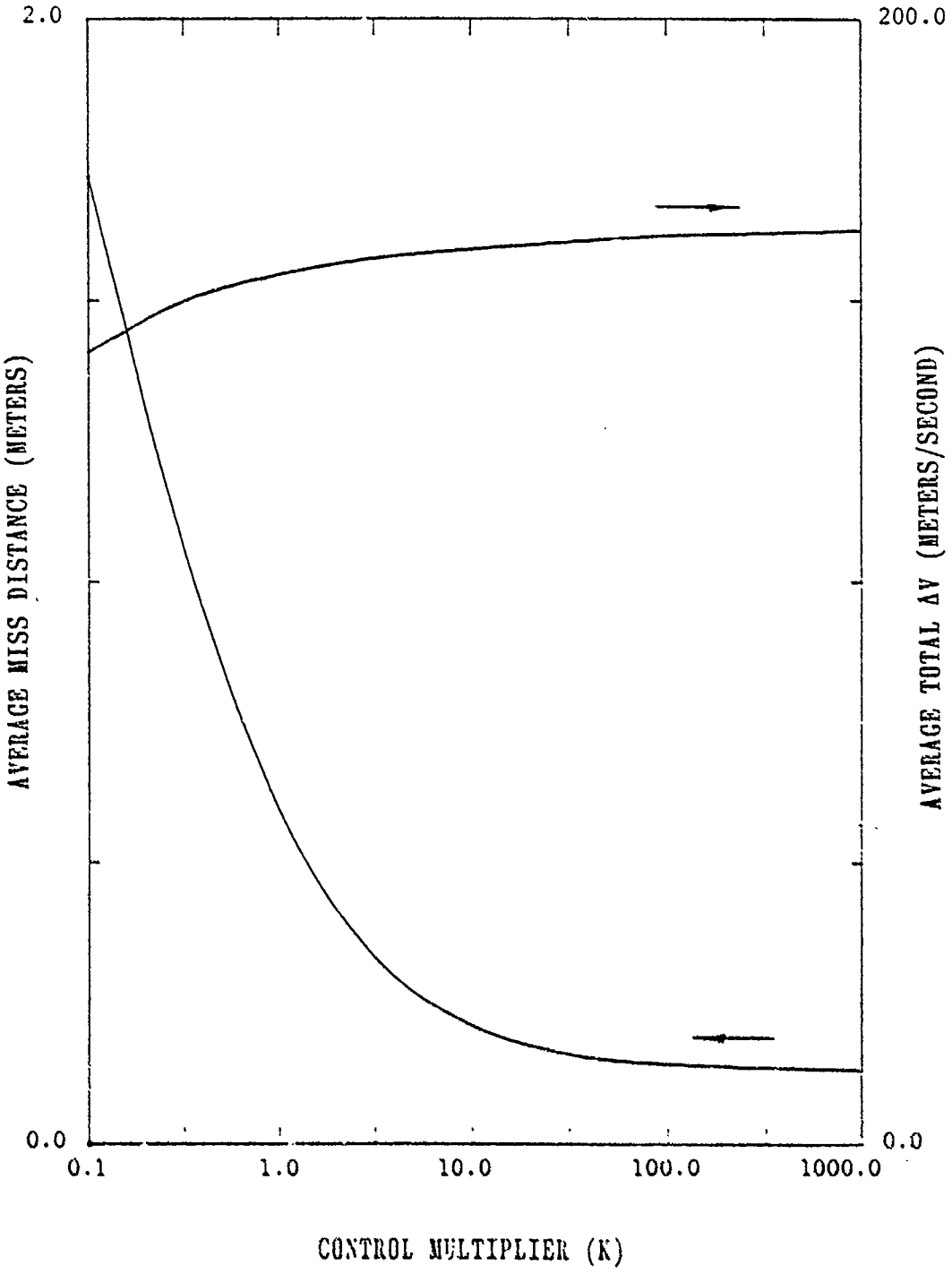

Figure 12-18. Performance of Dual Control for Case VI.

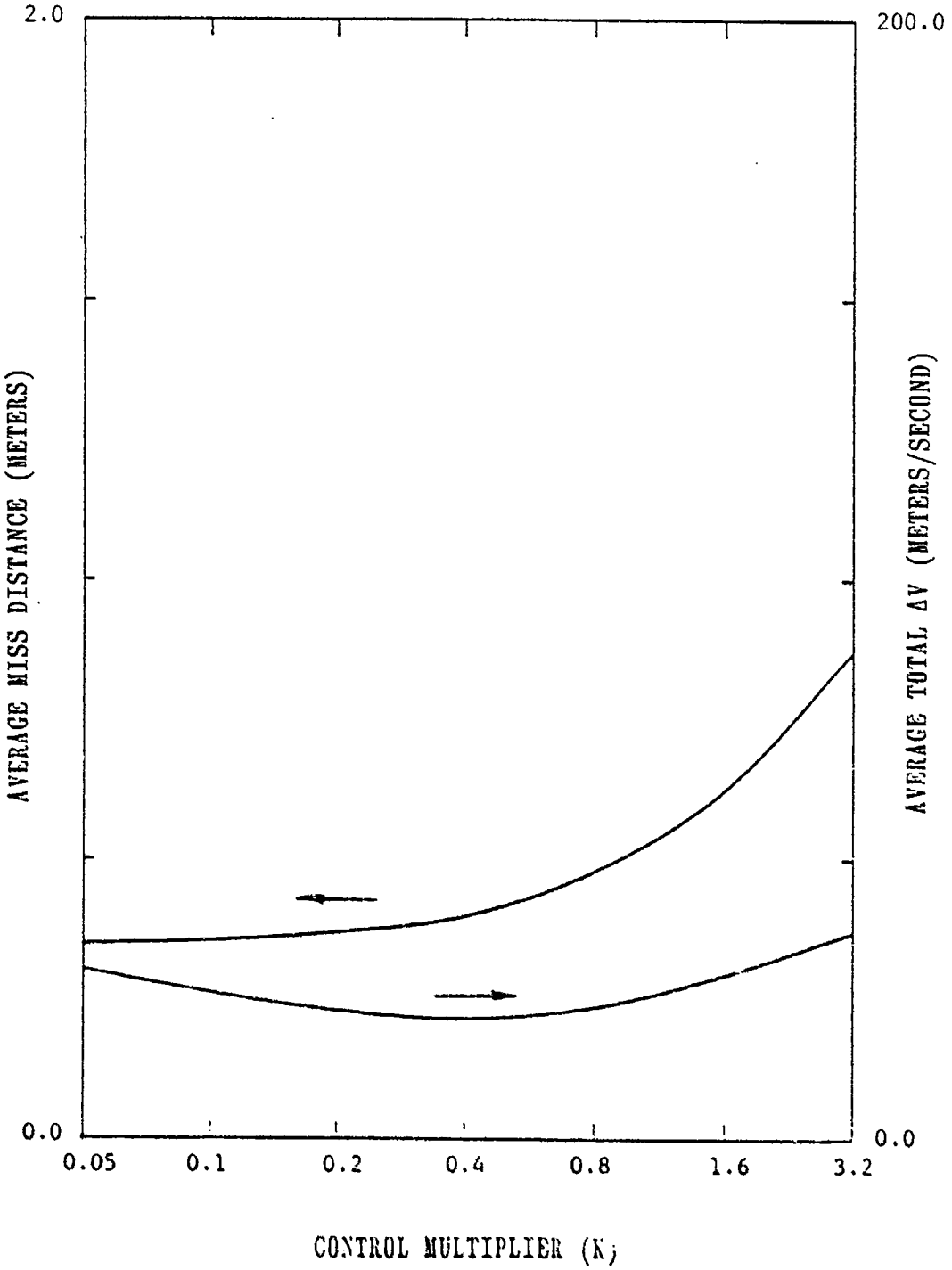

Figure 12-19. Performance of Certainty Control for Case I.

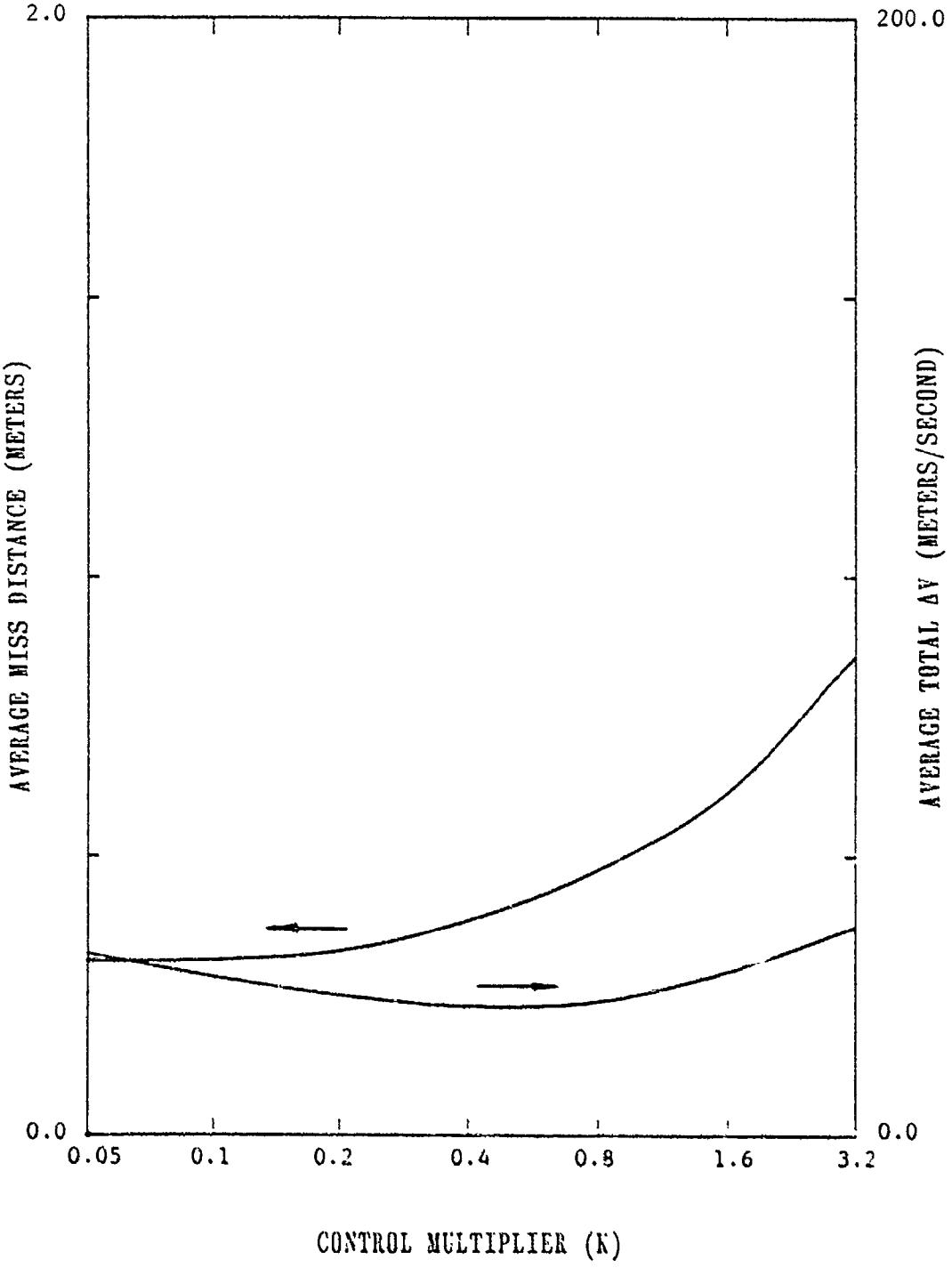

Figure 12-20. Performance of Certainty Control for Case II.

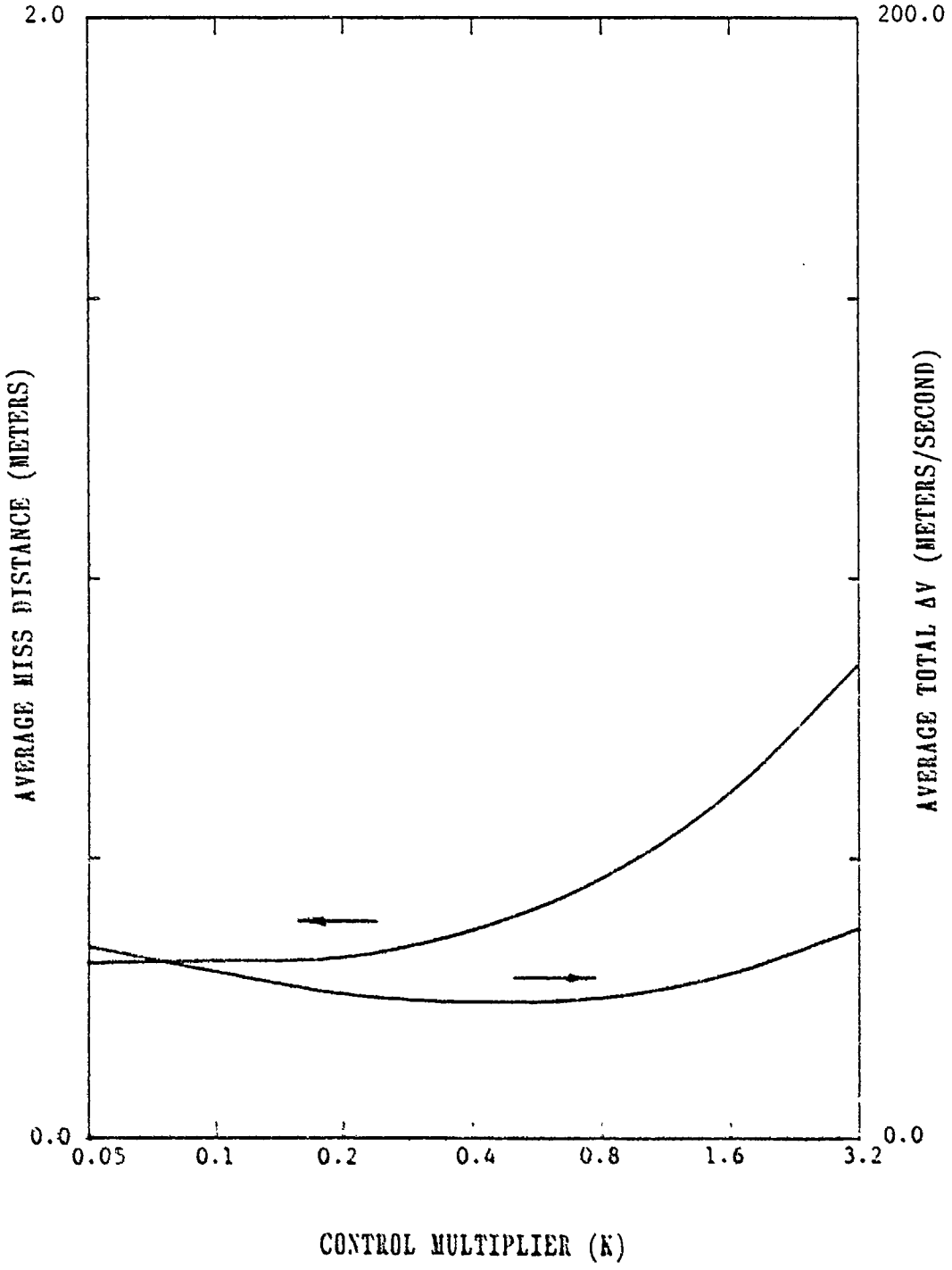

Figure 12-21. Performance of Certainty Control for Case III.

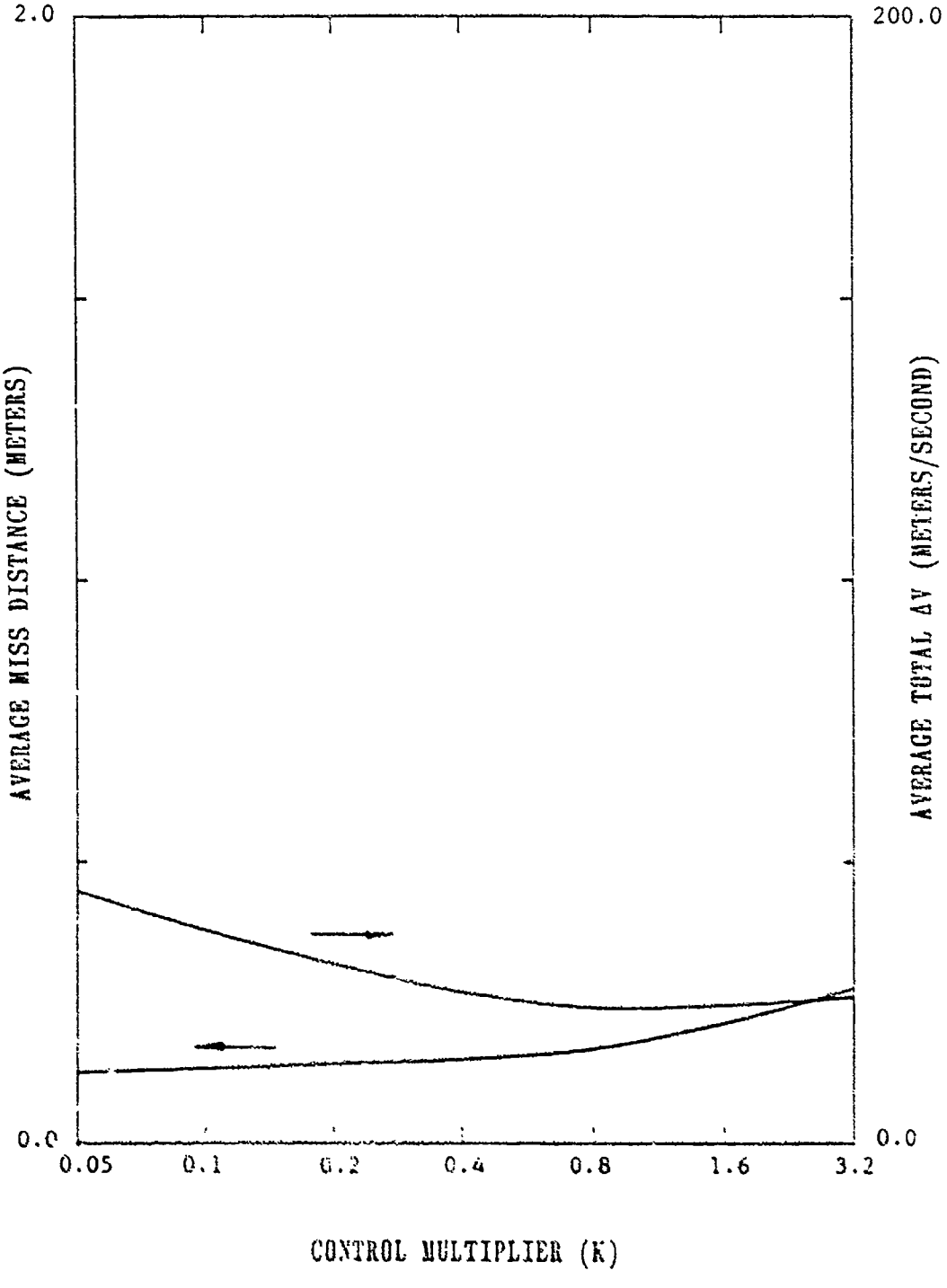

Figure 12-22. Performance of Certainty Control for Case IV.

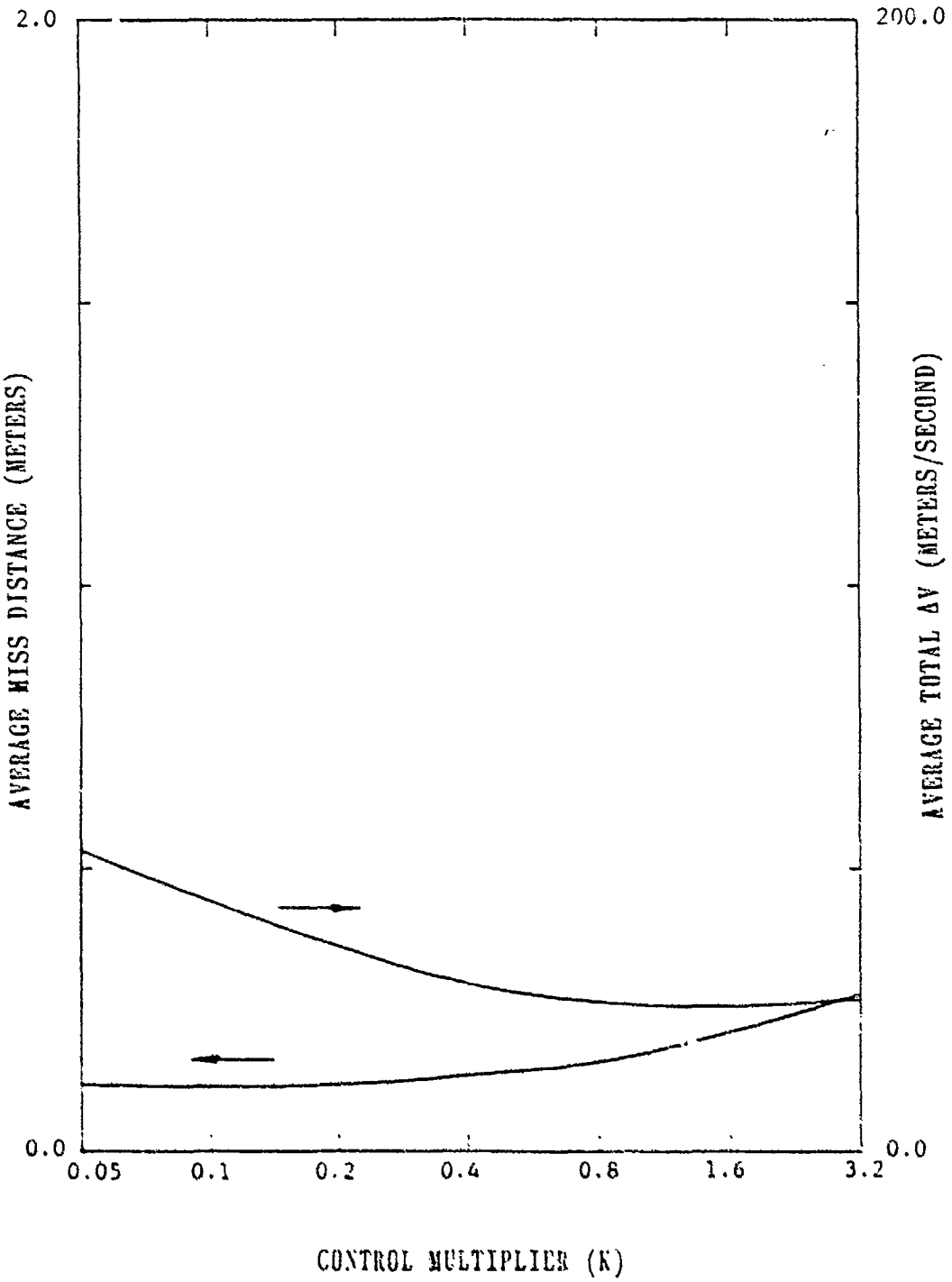

Figure 12-28. Performance of Certainty Control for Case V.

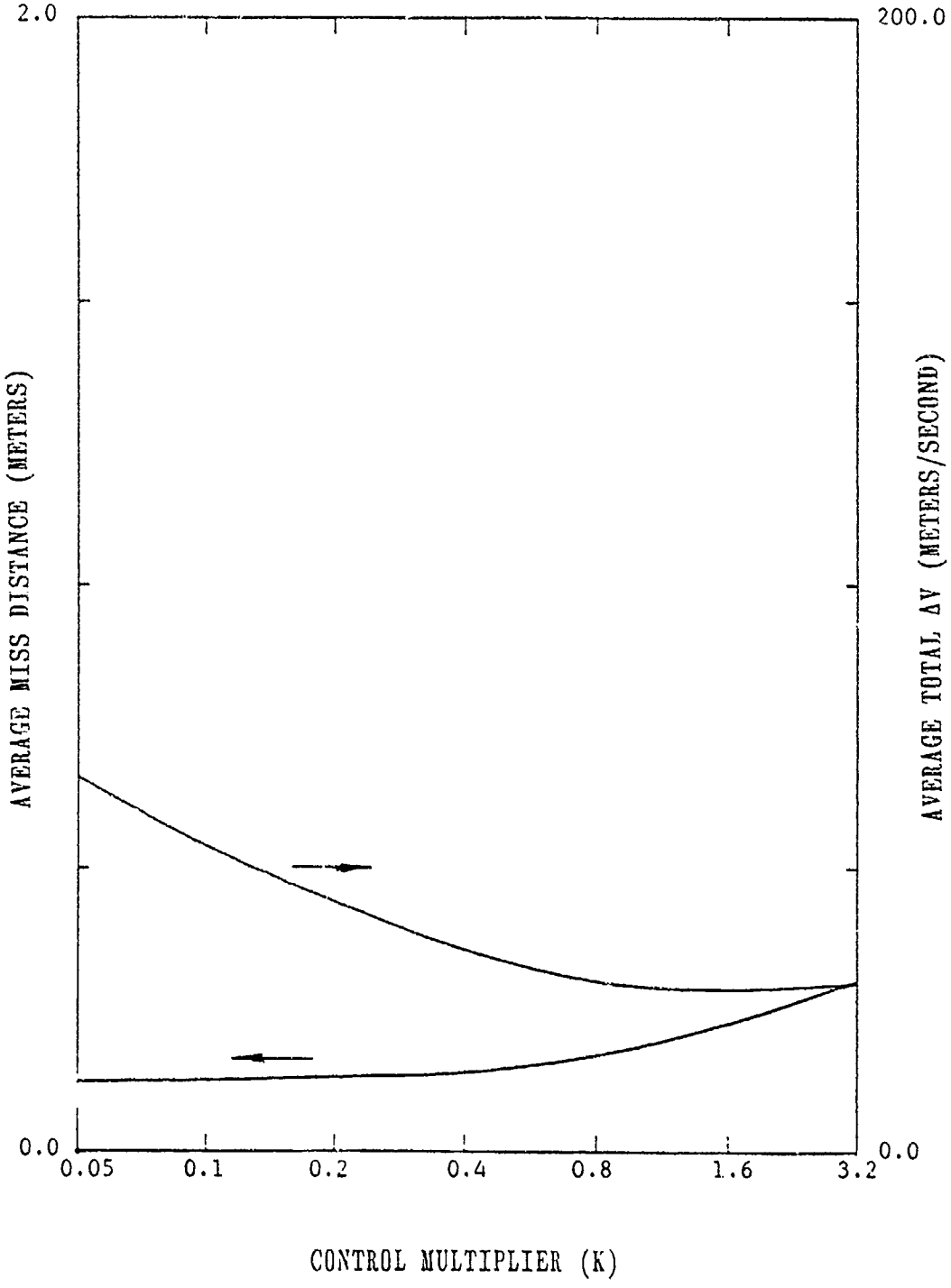

Figure 12-24. Performance of Certainty Control for Case VI.

Table 12-1. Case I Performance.

(Head On, In-plane Intercept)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.465	.230	81.93	5.64
PLAN B	.362	.173	88.29	7.00
PLAN C	.362	.174	90.76	6.86
OPTIMUM THRUST SPACING (ρ=1.75)	.363	.175	36.63	8.14
DUAL CONTROL (K=10)	.465	.230	81.93	5.64
CERTAINTY CONTROL (K=.4)	.399	.215	21.61	3.79
TRUTH WITH NOISE	.527	.264	81.44	6.76
TRUTH WITHOUT NOISE	0	NA	7.31	NA

Table 12-2. Case II Performance.

(Head On, 10° Out-of-Plane Intercept)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.502	.224	83.82	6.99
PLAN B	.360	.171	90.39	7.24
PLAN C	.360	.171	93.07	7.39
OPTIMUM THRUST SPACING (ρ=1.75)	.361	.171	37.19	8.50
DUAL CONTROL (K=10)	.502	.224	83.82	6.99
CERTAINTY CONTROL (K=.4)	.386	.191	23.21	4.18
TRUTH WITH NOISE	.545	.264	83.69	7.18
TRUTH WITHOUT NOISE	0	NA	7.54	NA

Tabel 12-3. Case III Performance.

(Head On, 20° Out-of-Plane Intercept)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.506	.225	90.92	7.51
PLAN B	.358	.168	97.53	7.93
PLAN C	.358	.168	99.97	7.89
OPTIMUM THRUST SPACING (ρ=1.75)	.358	.168	39.87	8.63
DUAL CONTROL (K=10)	.506	.225	90.92	7.51
CERTAINTY CONTROL (K=.4)	.372	.185	24.45	4.50
TRUTH WITH NOISE	.534	.293	90.77	7.55
TRUTH WITHOUT NOISE	0	NA	7.78	NA

Table 12-4. Case IV Performance.

(In-Plane Tail Chase)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.204	.105	110.87	9.89
PLAN B	.126	.058	113.72	9.81
PLAN C	.126	.057	111.82	9.25
OPTIMUM THRUST SPACING (ρ=1.75)	.124	.057	35.06	10.34
DUAL CONTROL (K=10)	.204	.105	110.87	9.89
CERTAINTY CONTROL (K=.4)	.150	.081	26.94	6.78
TRUTH WITH NOISE	.376	.221	105.71	8.93
TRUTH WITHOUT NOISE	0	NA	8.75	NA

Table 12-5. Case V Performance.

(10° Out-of-Plane Tail Chase)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.190	.100	129.61	13.29
PLAN B	.126	.061	132.65	13.32
PLAN C	.126	.061	129.47	12.26
OPTIMUM THRUST SPACING (ρ=1.75)	.126	.059	39.96	12.85
DUAL CONTROL (K=10)	.190	.100	129.61	13.29
CERTAINTY CONTROL (K=.4)	.136	.076	29.74	9.10
TRUTH WITH NOISE	.379	.204	123.57	11.61
TRUTH WITHOUT NOISE	0	NA	9.52	NA

Table 12-6. Case VI Performance.

(20° Out-of-Plane Tail Chase)

	MEAN MISS (METERS)	STANDARD DEVIATION (METERS)	MEAN ΔV (M/S)	STANDARD DEVIATION (M/S)
PLAN A (K=10)	.161	.079	160.42	14.89
PLAN B	.135	.064	162.73	14.85
PLAN C	.136	.064	157.91	13.50
OPTIMUM THRUST SPACING (ρ=1.75)	.135	.065	46.67	15.90
DUAL CONTROL (K=10)	.161	.079	160.42	14.89
CERTAINTY CONTROL (K=.8)	.171	.087	30.11	10.40
TRUTH WITH NOISE	.396	.233	151.56	13.37
TRUTH WITHOUT NOISE	0	NA	10.13	NA

As predicted, the dual control's performance is no better than the certainty equivalence formulation of Plan A. This is due to the fact that range is included as a measurement, causing the control to have virtually no effect on improving filter variance. Plan B is more accurate than Plan A, but more costly in energy. Again, this result is expected because the formulation of Plan B is based on infinite miss penalty ($K=\infty$) for Plan A. By optimally spacing the thrusts of Plan B, energy expenditure is considerably reduced with little or no sacrifice in accuracy.

Plan C is just as accurate as Plan B, with slightly greater cost resulting from large initial intercept range. This extra cost is attributed to the negligible gravity assumption used in the formulation of Plan C. For the smaller ranges associated with a tail chase, Plan C was actually less costly than Plan B.

In every case, certainty control yields the least energy expenditure. This result is not surprising, as the formulation of certainty control is based on reducing control energy in the presence of poor estimates. This form of control works best because filter variance is range dependent. As range decreases, the control constraint tightens, and accuracy increases. Therefore, less fuel is used when range is great, with refinements made as impact nears.

The last two entries (truth with and without noise) do not use splines. Trajectory changes are computed as outlined in

Chapter IV. This data is included as a baseline reference of performance.

Because ranging is an active and costly process, various tracking schemes were examined to determine if ranging is needed. This was easily done in the simulation because of the serial updating discussed in Chapter X. Using only line-of-sight measurement angles, all algorithms are less accurate and/or require more velocity changes. It is of interest to note that the dual control guidance scheme, true to its nature, did expend control energy to improve the estimate. The improvement was very slight because of the pursuer's speed and lateral thrusting limits. Allowing one range update at midcourse also proved costly for all guidance schemes.

An attempt was made to reduce the order of the filter in the hopes of reducing processing time. The result was a serious degradation of performance for all algorithms. The system model is very sensitive to the evader booster characteristics, A and $\dot{m}$, which are estimated by the eight state filter. Failure of the filtering process to refine initial booster estimates allows greater acceleration errors to be passed on to the evader estimates, significantly reducing end-game accuracy.

CHAPTER XIII

CONCLUSIONS AND AREAS OF FURTHER RESEARCH

In this research, six guidance schemes were examined to determine their capability to minimize lateral velocity changes of a hypervelocity orbital intercept vehicle. Proportional navigation, optimal control using certainty equivalance, dual control, control with optimum thrust spacing, and certainty control were all examined. Certainty control was shown to be the most energy efficient.

Certainty control constrains the final condition to a function of final estimator accuracy in the absence of updates. This general approach is not limited to hypervelocity vehicles, and would suggest other applications of this form to stochastically control intercepts.

This control requires a measure of final estimator accuracy which was achieved by running the Extended Kalman Filter forward to intercept time without updates. This time consuming process could be eliminated if filter variances could be estimated by some function (polynomial or otherwise). Also, the constraint multiplier was assumed constant for this formulation. Perhaps a multiplier that was range or time dependent would further reduce

interceptor thrusting.

In summary, the approach identified by this research not only improves the efficiency of hypervelocity intercept, but can be applied to a broad range of stochastic problems where control energy does not improve filter accurancy. It is also possible to combine the effects of dual and certainty control in certain cases by initially using dual control to improve estimator accuracy and then switching to certainty control. End-game accuracy may be improved by switching from certainty control to a certainty equivalence formulation just prior to impact.

BIBLIOGRAPHY

1. Guelman, M., "Qualitative Study of Proportional Navigation," IEEE Transactions on Aerospace and Electronic Systems, July 1971, pp. 337-343.

2. Guelman, M., "The Closed Form Solution of Pure Proportional Navigation," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-12, July 1976, pp. 472-482.

3. Sridhar, B. and N.K. Gupta, "Missile Guidance Laws Based on Singular Perturbation Methodology," Journal of Guidance and Control, Vol. 3, April 1980, pp. 158-166.

4. Tse, E., Bar-Shalom, Y., and L. Meier, III, "Wide Sense Adaptive Dual Control for Nonlinear Stochastic Systems," IEEE Transactions on Automatic Control, Vol. AC-18, No. 2, April 1973, pp. 98-108.

5. Tse, E., and Y. Bar-Shalom, "Adaptive Dual Control For Stochastic Nonlinear Systems with Free End-Time," IEEE Transactions on Automatic Control, October 1975, pp. 670-675.

6. Nesline, F.W., Wells, B.H., and P. Zachran, "Combined Optimal/Classic Approach to Robust Missile Autopilot Design," Journal of Guidance and Control, Vol. 4, No. 3, May-June 1981, pp.316-322.

7. Speyer, J.L., Hull, D.G., and C.Y. Tseng, "Estimation Enhancement by Trajectory Modulation for Homing Missiles," Journal of Guidance, Vol. 7, No. 2, March-April 1984, pp. 167-174.

8. Guelman, M., and J. Shinar, "Optimal Guidance Law in the Plane," Journal of Guidance, Vol. 7, No. 4, July-August 1984, pp. 471-476.

9. Tang, Y.M., and J.A. Borrie, "Missile Guidance Based on Kalman Filter Estimation of Target Maneuver," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-20, No. 6, Nov. 1984, pp. 736-741.

10. Yeuh, W.R., and C.F. Lin, "Optimal Controller for Homing Missile," Journal of Guidance, Vol. 8, No. 3, May-June 1985, pp. 408-411.

11. Lin, C.F., and S.P. Lee, "Robust Missile Autopilot Design Uing a Generalized Singular Optimal Control Technique," Journal of Guidance, Vol. 8, No. 4, July-August 1985, pp. 498-507.

12. Ashida, S., Howe, R.M., and N.X. Vinh, "Optimal Control of Air-Launched Homing Missiles Based on Realistic Performance Indices," Proceedings of the Conference on Aerospace Simulation II, Vol. 16, No. 2, Jan. 1986.

13. Lin, C.F., and L.L. Tsai, "Analytical Solution of Optimal Trajectory-Shaping Guidance," Journal of Guidance, Control, and Dynamics, Vol. 10, No. 1, Jan.-Feb. 1987, pp. 61-66.

14. Yang, C.D., and F.B. Yeh, "Closed-Form Solution for a Class of Guidance Laws," Journal of Guidance, Vol. 10, No. 4, July-August 1987, pp. 412-415.

15. Cherry, G.W., "A General, Explicit, Optimizing Guidance Law for Rocket-propelled Spaceflight," AIAA Paper 64-638, Aug. 1964.

16. Johnson, F.T., "Approximate Finite-Thrust Trajectory Optimization,," AIAA Journal, Vol. 7, June 1969, pp. 993-997.

17. Bate, R.R., Mueller, D.D., and J.E. White, Fundamentals of Astrodynamics, Dover Publications, New York, 1971.

18. Borisenko, I.I., and Y.P. Kulyabichev, "Algorithm for Optimization of the Solution of the Spacecraft Rendezvous Problem," Cosmic Research, Vol. 18, No. 3, May-June 1980, pp. 343-347.

19. Stuart, D.G., "A Simple Targeting Technique for Two-Body Spacecraft Trajectories," Journal of Guidance, Vol. 9, No. 1, Jan.-Feb. 1986, pp. 27-31.

20. Bhat, M.S., and S.K. Shrivastava, "An Optimal Q-Guidance Scheme for Satellite Launch Vehicles," Journal of Guidance, Vol. 10, No. 1, Jan.-Feb. 1987, pp. 53-60.

21. Menon, P.K.A., and A.J. Calise, "Interception, Evasion, Rendezvous and Velocity-to-be-Gained Guidance for Spacecraft," AIAA Paper 87-2318, Aug. 1987.

22. Dickmanns, F.D., and K.H. Wells, "Approximate Solution of Optimal Control Problems Using Third Order Hermite Polynomial Functions," Proceedings of the 6th Technical Conference on Optimization Techniques, Springer-Verlag, New York, IFIP-TC7, 1975.

23. Hargraves, C.R., and S.W. Paris, "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," Journal of Guidance, Vol. 10, No. 4, July-August 1987, pp.338-342.

24. Maron, M.J., Numerical Analysis: A Practical Approach, Macmillan Publishing Co., New York, 1982, pp. 177-182.

25. Bryson, A.E., and Y.C. Ho, Applied Optimal Control, Hemisphere Publishing Corp., Washington D.C., 1975, pp. 1-2.

26. Aoki, M., Optimization of Stochastic Systems, Academic Press Inc., New York, 1967, p. 10.

27. Lietmann, G., Optimization Techniques, Academic Press Dir. Inc., New York, 1962, Breakwell, J., Ch. 12, pp. 353-375.

28. Aoki, M., Optimization of Stochastic Systems, Academic Press Inc., New York, 1967, p. 3.

29. Bryson, A.E., and Y.C. Ho, Applied Optimal Control, Hemisphere Publishing Corp., Washington D.C., 1975, pp. 71-75.

30. Gelb, A., Applied Optimal Estimation, M.I.T. Press, Massachusetts, 1986, pp. 180-228.

31. Ho, Y.C., "On the Stochastic Approximation Method and Optimal Filtering Theory," Journal Mathematical Analysis and Applications, Vol. 6, 1963, pp. 152-154.

APPENDIX A

SPLINE APPROXIMATION ERRORS

Because the splines discussed in Chapter V are only approximations, there will be a small difference from the true trajectories modeled by them. To examine these errors, six figures are generated from a worst case scenario. Case I is considered the worst because of the high relative velocities. From this case profiles are generated for no velocity change, a velocity change of one meter per second, maximum ΔV_y, and maximum ΔV_z.

Figures A-1 and A-2 show no error at predicted impact time. This is expected because the splines are constrained to match final position and velocity. Figure A-3 reflects the error caused by a one meter per second in-plane velocity change decreasing as time-to-go decreases. Figure A-4 shows a similar effect for an out-of-plane velocity change. Figure A-5 shows the effect of maximum ΔV_y on trajectory error in the region of impact. Figure A-6 shows a similar effect for maximum out-of-plane thrusting.

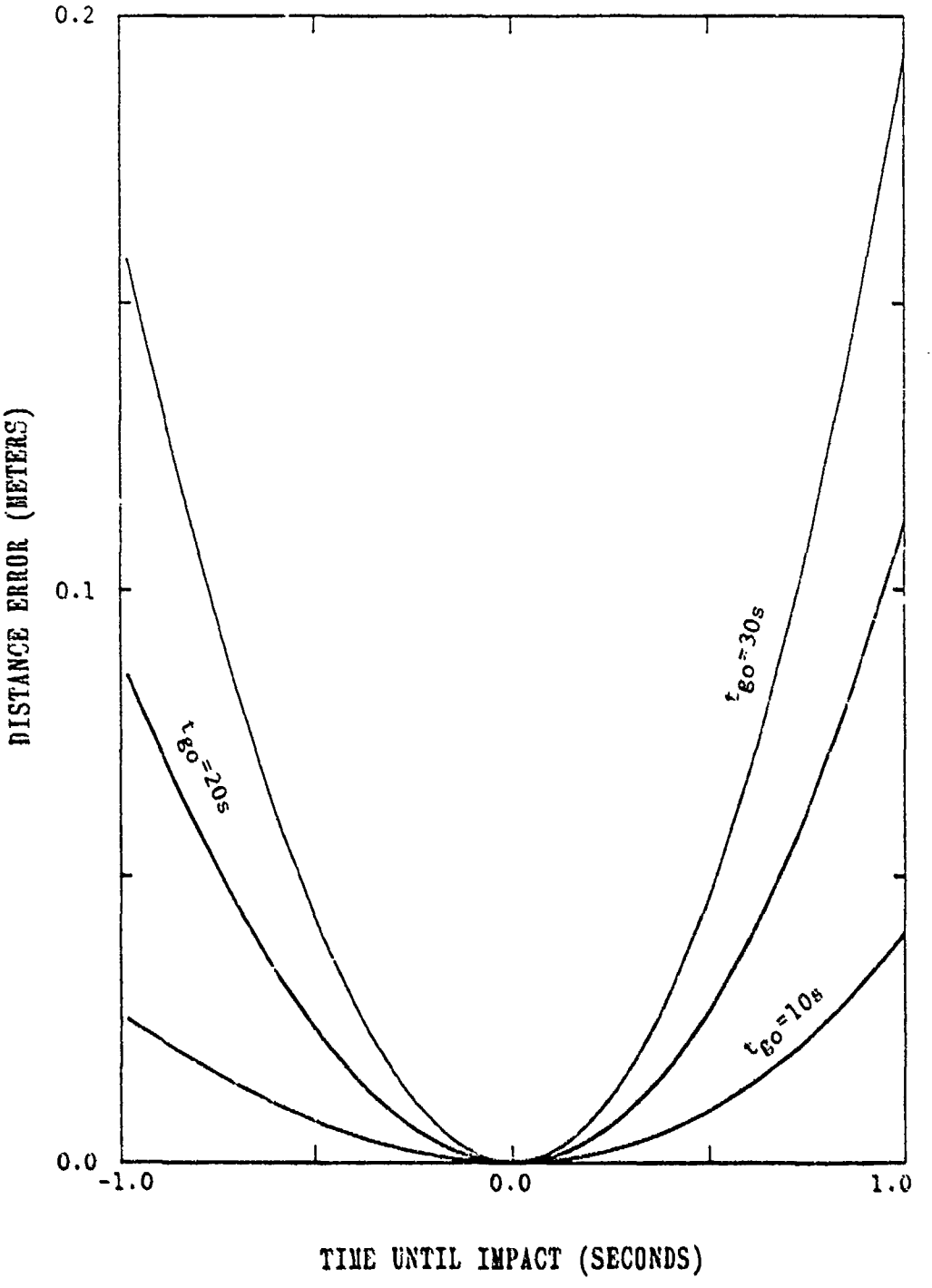

Figure A-1. Distance error of spline trajectory vs. time for zero velocity change.

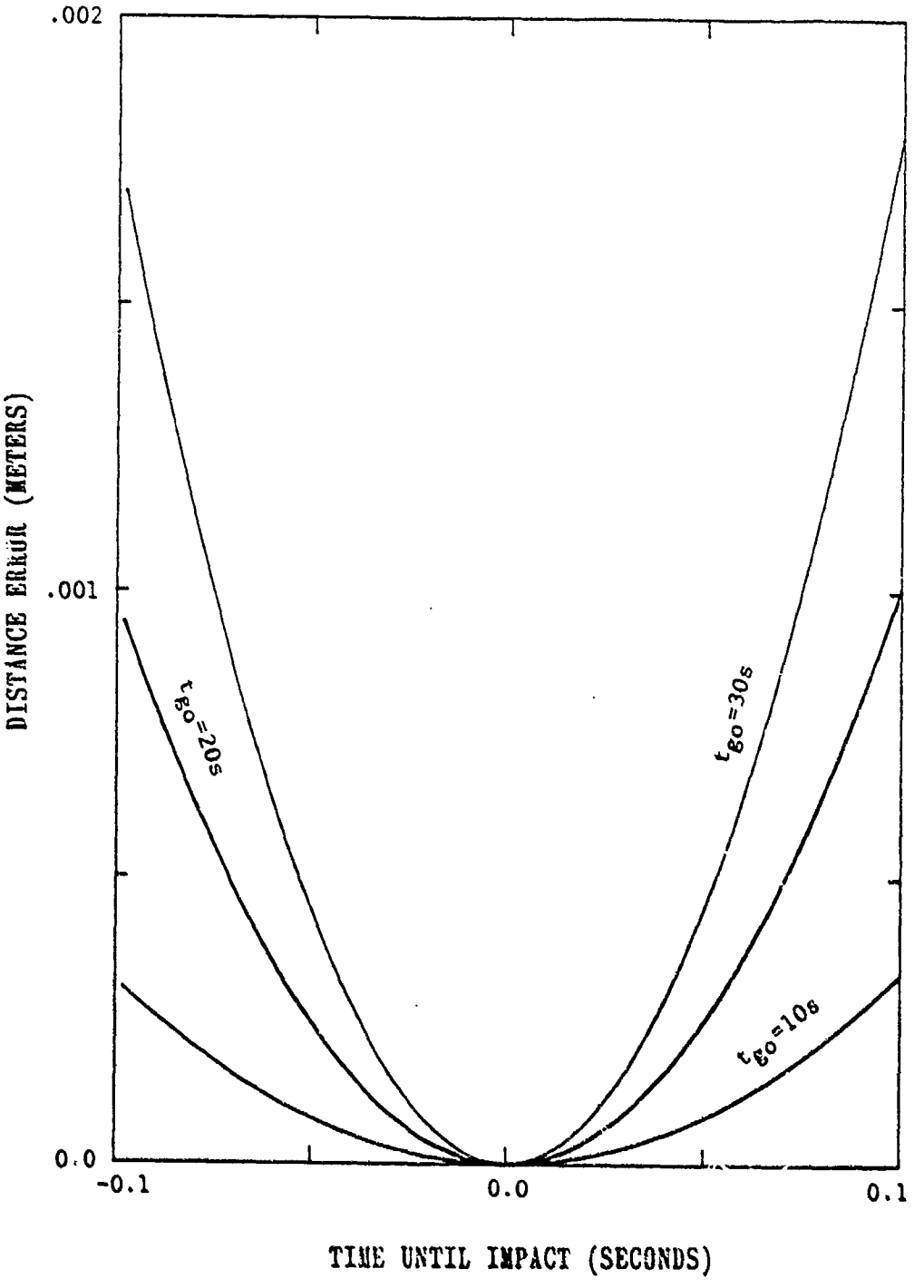

Figure A-2. Distance error of spline trajectory vs. time for zero velocity change.

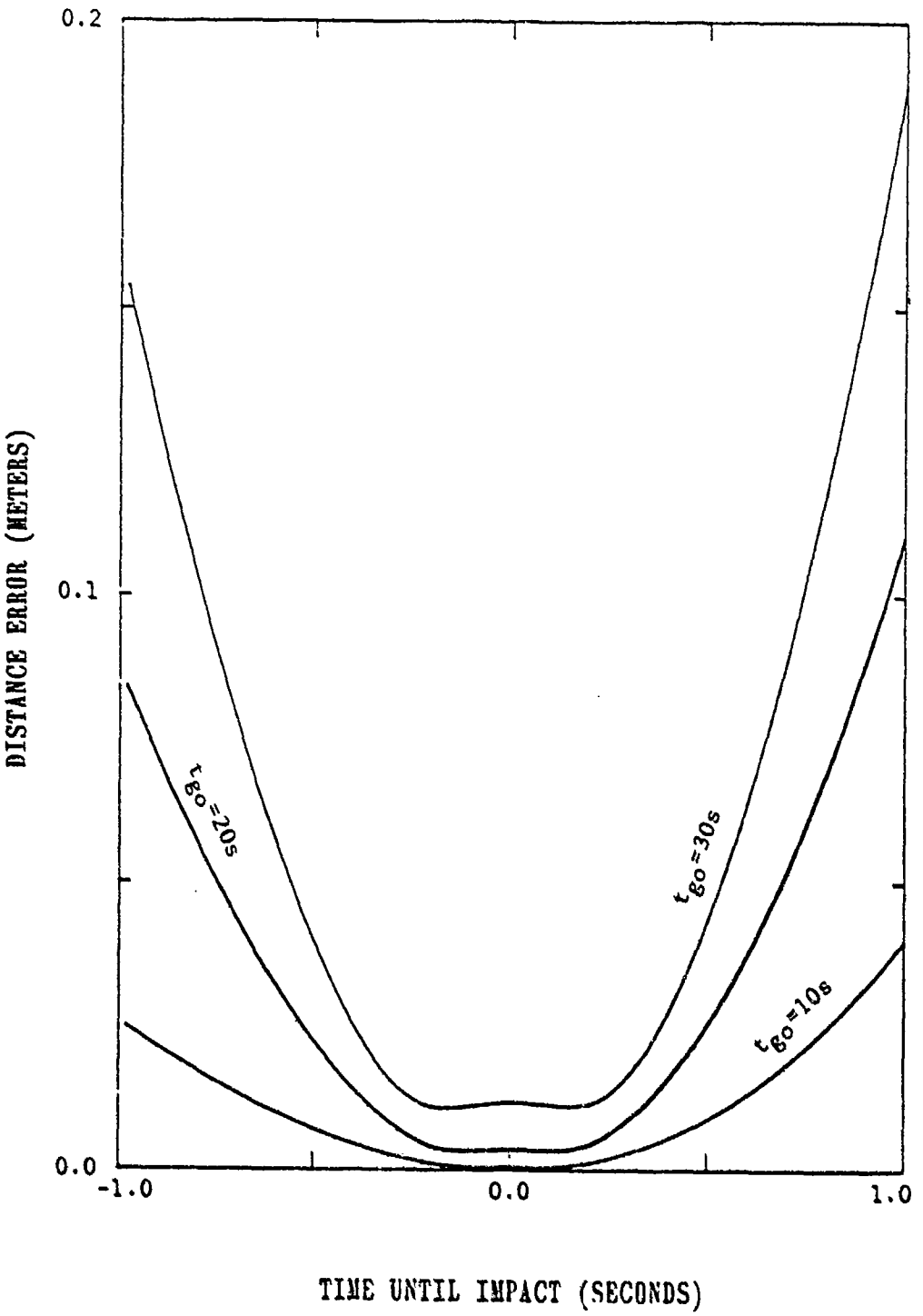

Figure A-3. Distance error of spline trajectory vs. time for ΔV_y = 1 m/s.

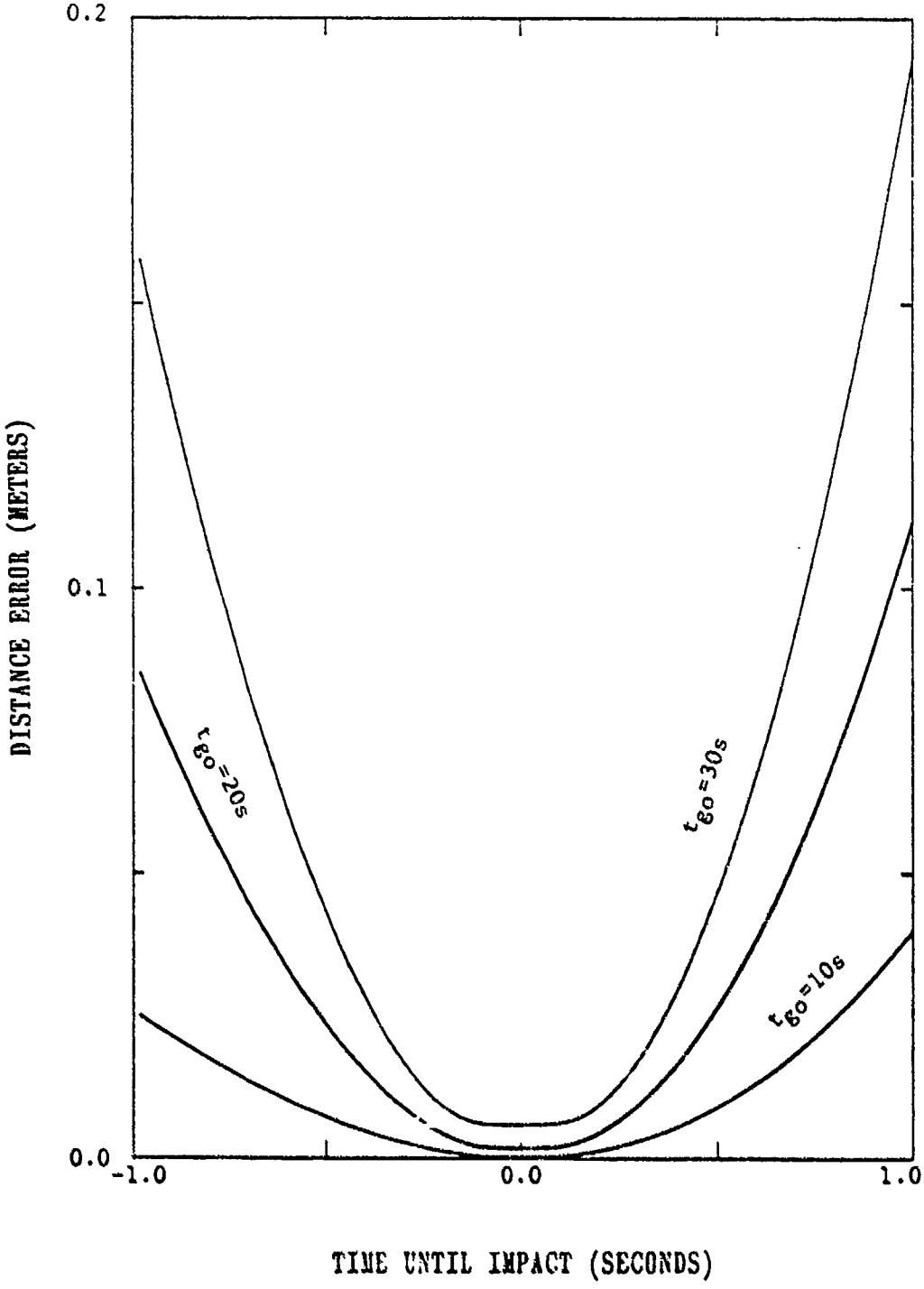

Figure A-4. Distance error of spline trajectory vs. time for ΔV_z = 1 m/s.

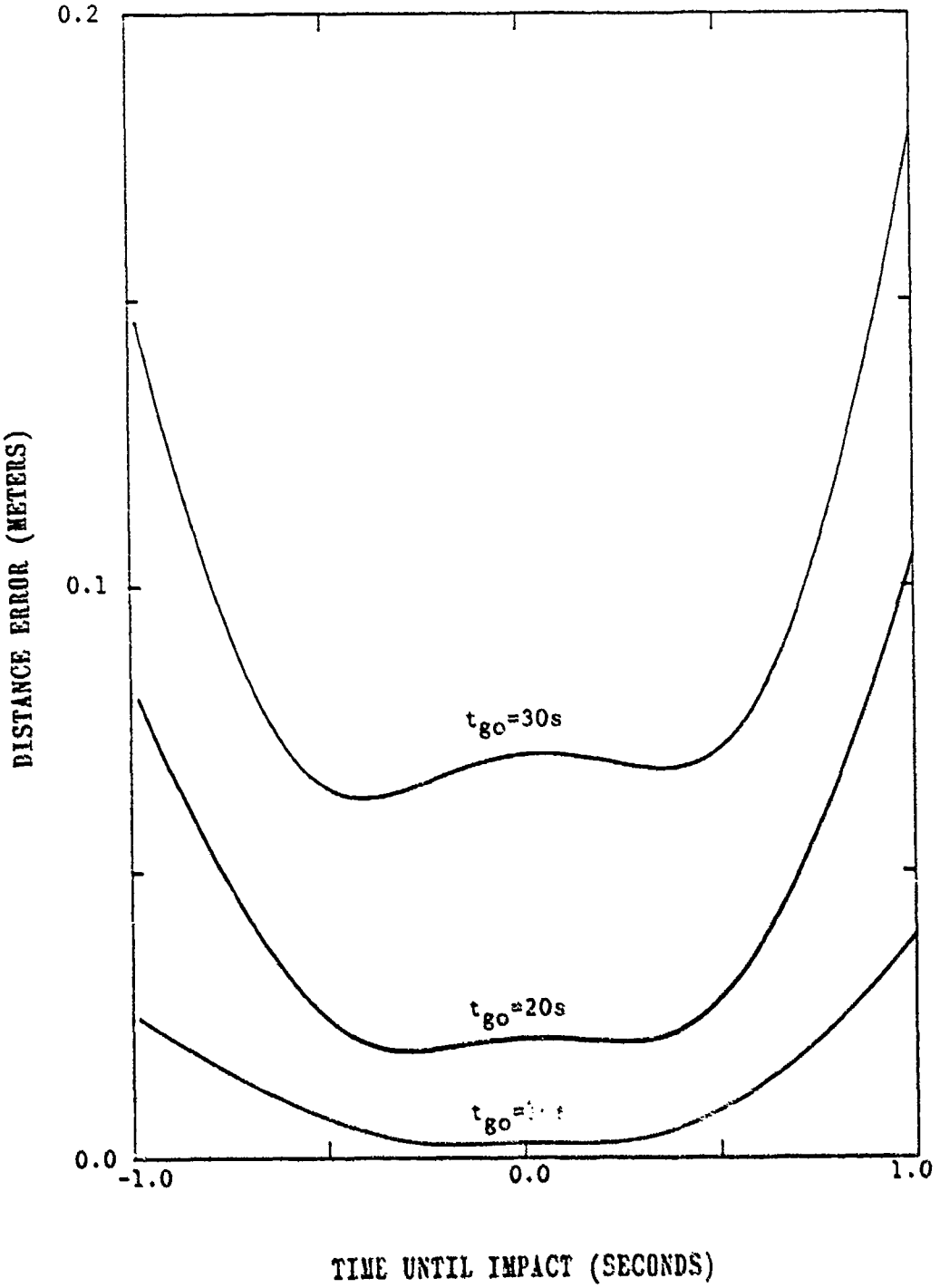

Figure A-5. Distance error of spline trajectory vs. time for maximum ΔV_y (ΔV_y = 6 m/s).

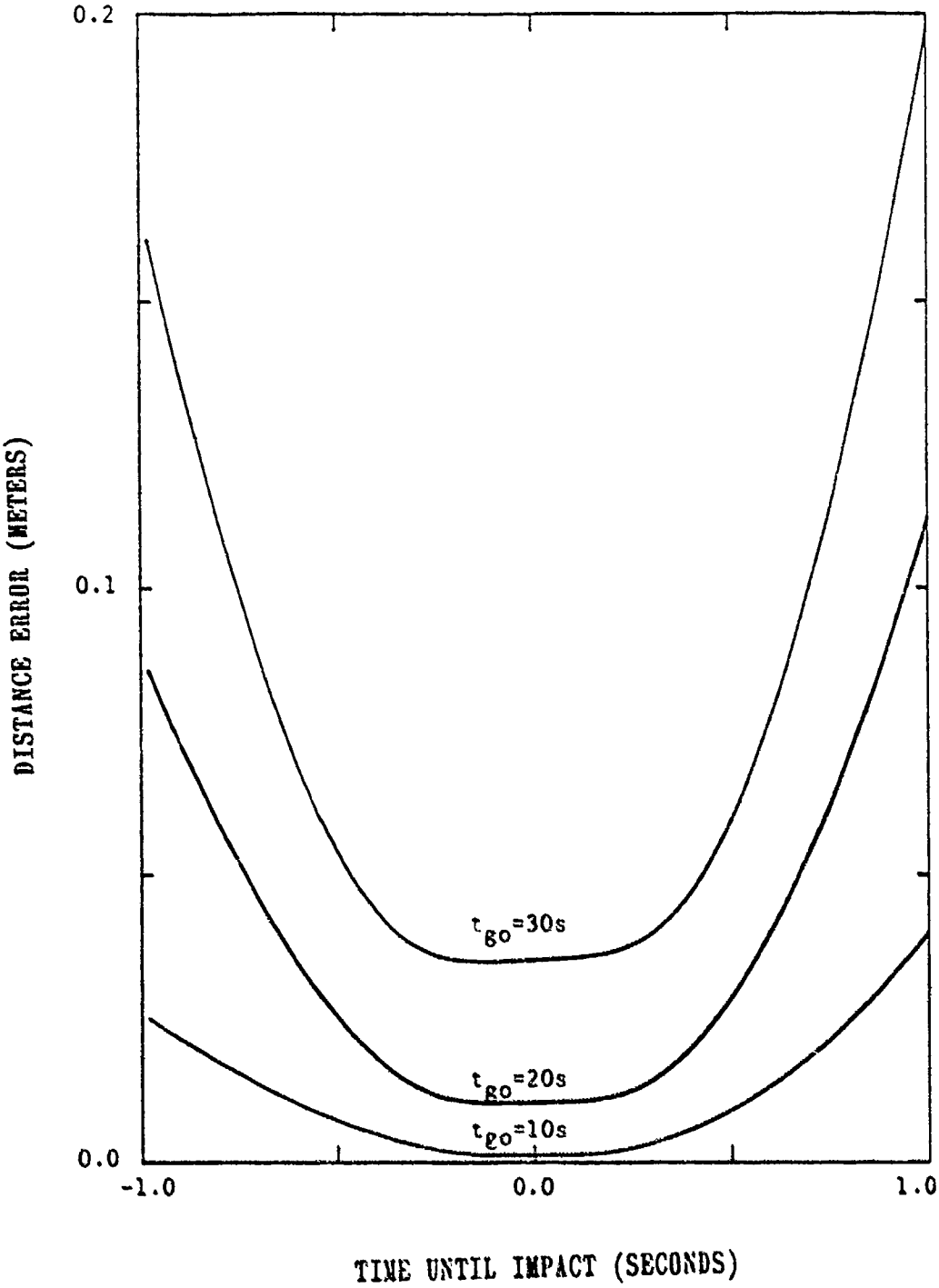

Figure A-6. Distance error of spline trajectory vs. time for maximum ΔV_z (ΔV_z = 6 m/s).

APPENDIX B

DERIVATION OF CERTAINTY CONTROL EQUATIONS

The dot terms for (8-15) are computed as follows:

$$\dot{x}_f = 3A_x t_{go}^2 + 2B_x t_{go} + C_x \tag{B-1}$$

$$\dot{y}_f = 3A_y t_{go}^2 + 2B_y t_{go} + C_y - \Delta V_y \tag{B-2}$$

$$\dot{z}_f = 3A_z t_{go}^2 + 2B_z t_{go} + C_z - \Delta V_z \tag{B-3}$$

$$\dot{\sigma}_{xf} = 3A_{\sigma x} t_{go}^2 + 2B_{\sigma x} t_{go} + C_{\sigma x} \tag{B-4}$$

$$\dot{\sigma}_{yf} = 3A_{\sigma y} t_{go}^2 + 2B_{\sigma y} t_{go} + C_{\sigma y} \tag{B-5}$$

$$\dot{\sigma}_{zf} = 3A_{\sigma z} t_{go}^2 + 2B_{\sigma z} t_{go} + C_{\sigma z} \tag{B-6}$$

The Jacobian matrix elements for (8-20) are:

$$J_{11} = \frac{\partial f_1}{\partial t_{go}} = f_2 \tag{B-7}$$

$$J_{12} = \frac{\partial f_1}{\partial \lambda} = y_f \frac{\partial y_f}{\partial \lambda} + z_f \frac{\partial z_f}{\partial \lambda} \tag{B-8}$$

$$\frac{\partial y_f}{\partial \lambda} = \frac{-y_s t_{go}^2}{(1 + \lambda t_{go}^2)^2} \tag{B-9}$$

$$\frac{\partial z_f}{\partial \lambda} = \frac{-z_s t_{go}^2}{(1 + \lambda t_{go}^2)^2} \tag{B-10}$$

$$J_{21} = \frac{\partial f_2}{\partial t_{go}} = x_f \ddot{x}_f + \dot{x}_f^2 + y_f \ddot{y}_f + \dot{y}_f^2 + z_f \ddot{z}_f + \dot{z}_f^2$$
$$- K[\sigma_{xf} \ddot{\sigma}_{xf} + \dot{\sigma}_{xf}^2 + \sigma_{yf} \ddot{\sigma}_{yf} + \dot{a}_{yf}^2 + \sigma_{zf} \ddot{\sigma}_{zf} + \dot{\sigma}_{zf}^2] \tag{B-11}$$

$$\ddot{x}_f = 6A_{\sigma x} t_{go} + 2B_x \tag{B-12}$$

$$\ddot{y}_f = 6A_y t_{go} + 2B_y - \lambda(\dot{y}_f t_{go} + y_f) \tag{B-13}$$

$$\ddot{z}_f = 6A_z t_{go} + 2B_z - \lambda(\dot{z}_f t_{go} + z_f) \tag{B-14}$$

$$\ddot{\sigma}_{xf} = 6A_{\sigma x} t_{go} + 2B_{\sigma x} \tag{B-15}$$

$$\ddot{\sigma}_{yf} = 6A_{\sigma y} t_{go} + 2B_{\sigma y} \tag{B-16}$$

$$\ddot{\sigma}_{zf} = 6A_{\sigma z} t_{go} + 2B_{\sigma z} \tag{B-17}$$

$$J_{22} = \frac{\partial f_2}{\partial \lambda} = \frac{\partial y_f}{\partial \lambda}\dot{y}_f + y_f\frac{\partial \dot{y}_f}{\partial \lambda} + \frac{\partial z_f}{\partial \lambda}\dot{z}_f + z_f\frac{\partial \dot{z}_f}{\partial \lambda} \tag{B-18}$$

$$\frac{\partial \dot{y}_f}{\partial \lambda} = -\frac{y_s t_{go}}{(1 + \lambda t_{go}^2)^2} \tag{B-19}$$

$$\frac{\partial \dot{z}_f}{\partial \lambda} = -\frac{z_s t_{go}}{(1 + \lambda t_{go}^2)^2} \tag{B-20}$$

APPENDIX C

EXTENDED KALMAN FILTER EQUATIONS

The EKF states are defined from (3-5) through (3-11) as follows:

$$x_1 = x_E - x_p \tag{C-1}$$

$$x_2 = \dot{x}_E - \dot{x}_p \tag{C-2}$$

$$x_3 = y_E - y_p \tag{C-3}$$

$$x_4 = \dot{y}_E - \dot{y}_p \tag{C-4}$$

$$x_5 = z_E - z_p \tag{C-5}$$

$$x_6 = \dot{z}_E - \dot{z}_p \tag{C-6}$$

$$x_7 = A \tag{C-7}$$

$$x_8 = \dot{m} \tag{C-8}$$

Determining the F matrix components from (10-7) yields

$$E = \sqrt{x_E^2 + y_E^2 + z_E^2} \tag{C-9}$$

$$\dot{E} = \sqrt{\dot{x}_E^2 + \dot{y}_E^2 + \dot{z}_E^2} \tag{C-10}$$

$$F_{12} = F_{34} = F_{56} = 1 \tag{C-11}$$

$$F_{21} = \frac{-\mu}{E^3} + \frac{3x_E^2\mu}{E^5} \tag{C-12}$$

$$F_{22} = \frac{-x_E^2 A}{E^3} + \frac{A}{E} \tag{C-13}$$

$$F_{23} = F_{41} = \frac{3x_E y_E \mu}{E^5} \tag{C-14}$$

$$F_{24} = F_{42} = \frac{-\dot{x}_E \dot{y}_E A}{E^3} \tag{C-15}$$

$$F_{25} = F_{61} = \frac{3x_E z_E \mu}{E^5} \tag{C-16}$$

$$F_{26} = F_{62} = \frac{-\dot{x}_E \dot{z}_E A}{E^3} \tag{C-17}$$

$$F_{27} = \frac{\dot{x}_E}{E} \tag{C-18}$$

$$F_{43} = \frac{-\mu}{E^3} + \frac{3y_E^2\mu}{E^5} \tag{C-19}$$

$$F_{44} = \frac{-\dot{y}_E^2 A}{\dot{E}^3} + \frac{A}{\dot{E}} \qquad (C-20)$$

$$F_{45} = F_{63} = \frac{3y_E z_E \mu}{E^5} \qquad (C-21)$$

$$F_{46} = F_{64} = \frac{-\dot{y}_E \dot{z}_E A}{\dot{E}^3} \qquad (C-22)$$

$$F_{47} = \frac{\dot{y}_E}{\dot{E}} \qquad (C-23)$$

$$F_{65} = \frac{-\mu}{E^3} + \frac{3z_E^2 \mu}{E^5} \qquad (C-24)$$

$$F_{66} = \frac{-\dot{z}_E^2 A}{\dot{E}^3} + \frac{A}{\dot{E}} \qquad (C-25)$$

$$F_{67} = \frac{\dot{z}_E}{\dot{E}} \qquad (C-26)$$

$$F_{77} = \dot{m} \qquad (C-27)$$

$$F_{78} = A \qquad (C-28)$$

$$F_{88} = 2\dot{m} \quad . \tag{C-29}$$

All other elements are zero.

The measurements of range and line-of sight angles are:

$$z_{1k} = \sqrt{x_1^2 + x_3^2 + x_5^2} + V_{Rk} \tag{C-30}$$

$$z_{2k} = TAN^{-1}(x_5/x_1) + V_{\theta k} \tag{C-31}$$

$$z_{3k} = TAN^{-1}(x_3/x_1) + V_{\gamma k} \tag{C-32}$$

The H_k vectors for serial update from (10-8) are:

$$R = \sqrt{x_1^2 + x_3^2 + x_5^2} \tag{C-33}$$

$$H_{1k1} = x_1/R \tag{C-34}$$

$$H_{1k3} = x_3/R \tag{C-35}$$

$$H_{1k5} = x_5/R \tag{C-36}$$

$$H_{2k1} = \frac{-x_5}{(x_1^2 + x_5^2)} \tag{C-37}$$

$$H_{2k5} = \frac{x_1}{(x_1^2 + x_5^2)} \tag{C-38}$$

$$H_{3k1} = \frac{-x_3}{(x_1^2 + x_3^2)} \tag{C-39}$$

$$H_{3k3} = \frac{x_1}{(x_1^2 + x_3^2)} \tag{C-40}$$

All other elements are zero.

APPENDIX D

COMPUTER SIMULATION PROGRAM

Contained here are the routines used to simulate the guidance algorithms of Chapters IV through IX. The program (main code) is separate from the supporting routines (subroutines) with the following labeling:

1. KEWSIM - This is the main program for the hypervelocity orbital intercept.

2. TOOL 1 - Contained here are the subroutines needed for orbit propagation.

3. TOOL 2 - The coordinate transformation matrix subroutines are found here.

4. TOOL 3 - The Extended Kalman Filter subroutines are kept here.

5. TOOL 4 - The subroutines for all the guidance algorithms plus the truth model are here.

The source code for the above can be found on the following pages. All the code is written in FORTRAN 77.

```
C     PROGRAM KEWSIM

C  THIS PROGRAM IS A HYPERVELOCITY ORBITAL INTERCEPT
C  SIMULATION THAT FINDS THE VELOCITY CHANGES WITH AN
C  EIGHT, SIX OR THREE STATE EXTENDED KALMAN FILTER
C  AND THREE MEASUREMENTS.

C    LINK AS FOLOWS:
C      LINK KEWSIM,TOOL1,TOOL2,TOOL3,TOOL4

C  PROGRAM DICTIONARY

C      A          EVADER ACCELERATION DUE TO THRUSTING
C      AD         DUMMY ACCELERATION
C      ASIGX      SIGMA X AXIS SPLINE COEFFICIENT OF T**3
C      ASIGY      SIGMA Y AXIS SPLINE COEFFICIENT OF T**3
C      ASIGZ      SIGMA Z AXIS SPLINE COEFFICIENT OF T**3
C      AT         DUMMY ACCELERATION
C      AX         X AXIS SPLINE COEFFICIENT OF T**3
C      AY         Y AXIS SPLINE COEFFICIENT OF T**3
C      AZ         Z AXIS SPLINE COEFFICIENT OF T**3
C      BSIGX      SIGMA X AXIS SPLINE COEFFICIENT OF T**2
C      BSIGY      SIGMA Y AXIS SPLINE COEFFICIENT OF T**2
C      BSIGZ      SIGMA Z AXIS SPLINE COEFFICIENT OF T**2
C      BX         X AXIS SPLINE COEFFICIENT OF T**2
C      BY         Y AXIS SPLINE COEFFICIENT OF T**2
C      BZ         Z AXIS SPLINE COEFFICIENT OF T**2
C      COUNT      ITERATION FINAL COUNT
C      COV        COVARIANCE MATRIX
C      CVD        COVARIANCE MATRIX TRACE ELEMENTS
C      CVDD       DUMMY COVARIANCE MATRIX
C      CVDUAL     DUMMY COVARIANCE MATRIX
C      CSIGX      SIGMA X AXIS SPLINE COEFFICIENT OF T
C      CSIGY      SIGMA Y AXIS SPLINE COEFFICIENT OF T
C      CSIGZ      SIGMA Z AXIS SPLINE COEFFICIENT OF T
C      CX         X AXIS SPLINE COEFFICIENT OF T
C      CY         Y AXIS SPLINE COEFFICIENT OF T
C      CZ         Z AXIS SPLINE COEFFICIENT OF T
C      DDX        CHANGE IN X VELOCITY (INERTIAL FRAME)
C      DDY        CHANGE IN Y VELOCITY (INERTIAL FRAME)
C      DDZ        CHANGE IN Z VELOCITY (INERTIAL FRAME)
C      DELTAY     CHANGE IN Y VELOCITY (BODY FRAME)
C      DELTAZ     CHANGE IN Z VELOCITY (BODY FRAME)
C      DH         DUMMY STEP SIZE
C      DSIGX      SIGMA X AXIS SPLINE COEFFICIENT
C      DSIGY      SIGMA Y AXIS SPLINE COEFFICIENT
C      DSIGZ      SIGMA Z AXIS SPLINE COEFFICIENT
C      DTGO       DUMMY TIME-TO-GO
C      DUM        DUMMY VARIABLE
C      DV         INCREMENTAL VELOCITY CHANGE
C      DX         X AXIS SPLINE COEFFICIENT
C      DY         Y AXIS SPLINE COEFFICIENT
C      DZ         Z AXIS SPLINE COEFFICIENT
C      FILTER     NUMBER OF FILTER STATES
```

```
C     GAMMA     OBSERVED LINE-OF-SIGHT ANGLE (IN PLANE)
C     G2        GAUSSIAN POINT
C     H         STEP SIZE
C     I         ITERATION COUNTER
C     J         ITERATION COUNTER
C     JUP       UPPER LIMIT ON 'J' ITERATION COUNTER
C     K         CONSTRAINT/COST FUNCTION MULTIPLIER
C     KDEVF     CONSTRAINT BASED ON FINAL COVARIANCE
C     KFLAG     KALMAN GAIN CONVERGENCE FLAG
C     MAXDV     MAXIMUM INCREMENTAL VELOCITY CHANGE
C     MAXG      MAXIMUM THRUST ACCELERATION (G FORCES)
C     MDOT      UNITIZED MASS FLOW RATE OF EVADER
C     MDOTD     DUMMY MASS FLOW RATE OF EVADER
C     MDOTT     DUMMY MASS FLOW RATE OF EVADER
C     MEAN      MEAN OF MEASUREMENT RESIDUALS
C     MINDV     MINIMUM INCREMENTAL VELOCITY CHANGE
C     MING      MINIMUM THRUST ACCELERATION (G FORCES)
C     MISS2     ESTIMATED MISS DISTANCE SQUARED
C     OPT       OPTION
C                 1 - WITHOUT KALMAN FILTER
C                 2 - WITH KALMAN FILTER
C                 3 - WITH KALMAN FILTER + PRINTOUT
C     PLAN      PLAN OPTION
C                 1 - PLAN A
C                 2 - PLAN B
C                 3 - PLAN C
C                 4 - DUAL CONTROL
C                 5 - CERTAINTY CONTROL
C                 6 - TRUTH MODEL
C     Q         PROPAGATION NOISE VARIANCE
C     RANGE     RANGE MEASUREMENT OF EVADER FROM PURSUER
C     RES       MEASUREMENT RESIDUALS
C     RHO       CONTROL EFFECTIVENESS RATIO
C     R3        MEASUREMENT NOISE VARIANCE
C     SEED      RANDOM NUMBER SEED
C     SFILTR    SIMULATION FILTER
C     SFLAG     SEARCH CONVERGENCE FLAG
C     SIGMAM    STANDARD DEVIATION OF MEASUREMENTS
C     SIGT0     INITIAL X,Y,Z DEVIATIONS AND THEIR RATES
C     SIGTF     FINAL X,Y,Z DEVIATIONS AND THEIR RATES
C     SIMCNT    SIMULATION COUNTER
C     SKFLAG    SIMULATION KALMAN GIAN CONVERGENCE FLAG
C     SNUM      SIMULATION NUMBER
C     SPLAN     SIMULATION PLAN
C     SRANGE    SIMULATION RANGE
C     SSFLAG    SIMULATION SEARCH CONVERGENCE FLAG
C     SVTOT     SIMULATION TOTAL VELOCITY CHANGE
C     SW        INTEGER SWITCH FOR FUNCTION 'GAUSS'
C     T         TIME
C     TD        DUMMY TIME
C     TGO       TIME-TO-GO (UNTIL IMPACT)
C     THETA     OBSERVED LOS ANGLE (OUT OF PLANE)
C     TMAT      TRANSFORMATION MATRIX
C     TOL       RANGE TOLERANCE FOR SEARCH ROUTINE
```

```
C      TSTART     START TIME FOR CONTROL
C      UPDATE     UPDATE FLAG FOR EKF
C                   0 - NO UPDATE
C                   1 - UPDATE
C                   2 - UPDATE WITH RESIDUALS SET TO ZERO
C      VAR        VARIANCE OF MEASUREMENT RESIDUALS
C      VEL        EVADER VELOCITY
C      VTOT       TOTAL VELOCITY CHANGE
C      XDUAL      DUMMY XHAT VECTOR
C      XDUALD     DUMMY XHAT VECTOR
C      XE         STATE VECTOR OF EVADER
C      XED        DUMMY STATE VECTOR OF EVADER
C      XEDD       DUMMY STATE VECTOR OF EVADER
C      XEEST      ESTIMATED STATE VECTOR OF EVADER
C      XET        TRANSFORMED STATE VECTOR OF EVADER
C      XHAT       ESTIMATED STATES
C                   1 - RELATIVE X POSITION
C                   2 - RELATIVE X VELOCITY
C                   3 - RELATIVE Y POSITION
C                   4 - RELATIVE Y VELOCITY
C                   5 - RELATIVE Z POSITION
C                   6 - RELATIVE Z VELOCITY
C                   7 - A
C                   8 - MDOT
C      XP         STATE VECTOR OF PURSUER
C      XPD        DUMMY STATE VECTOR OF PURSUER
C      XPDD       DUMMY STATE VECTOR OF PURSUER
C      XPEST      ESTIMATED STATE VECTOR OF PURSUER
C      XPT        TRANSFORMED STATE VECTOR OF PURSUER
C      XR         RELATIVE EVADER STATE VECTOR

C  DECLARE VARIABLES
       REAL*8 A,MDOT,T,TIME,H,XE(6),XP(6),XED(6),XPD(6)
       REAL*8 TGO,MING,RES(3),VAR(3),MEAN(3),XHAT(8)
       REAL*8 DELTAY,DELTAZ,DV,K,DH,TD,TMAT(3,3),MAXG
       REAL*8 AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,CZ,DZ,G2
       REAL*8 DDW,DDY,DDZ,XET(6),XPT(6),SIGMAM(3),MISS2
       REAL*8 COV(8,8),Q(8,8),R3(3,3),XEEST(6),XPEST(6)
       REAL*8 MAXDV,MINDV,VTOT,XR(6),RANGE,THETA,GAMMA
       REAL*8 MDOTD,MDOTT,SIGT0(6),SIGTF(6)
       REAL*8 GAUSS,AD,AT,TOL,DUM,TSTART,KDEVF,RHO
       REAL*8 SRANGE,SVTOT,ASIGZ,BSIGZ,CSIGZ,DSIGZ
       REAL*8 ASIGX,BSIGX,CSIGX,DSIGX,ASIGY,BSIGY,CSIGY
       REAL*8 DSIGY,CVD(6),XPDD(6),CVDD(8,8),XEDD(6)
       REAL*8 XDUAL(8),XDUALD(8),CVDUAL(8,8),DTGO
       INTEGER I,J,JUP,COUNT,SIMCNT,SEED,OPT,SW,PLAN
       INTEGER SNUM,UPDATE,SSFLAG
       INTEGER KFLAG,SFLAG,FILTER,SPLAN,SFILTR,SKFLAG
```

```
C  ********************
C  * INITIALIZATIONS *
C  ********************


C  READ IN INITIAL CONDITIONS FOR DYNAMICS
1     FORMAT(2X,3F14.3)
6     FORMAT(2X,F8.2)
8     FORMAT(2X,F9.5)

      OPEN(UNIT=2,NAME='[ENGR.THESIS.SALFANO]INIT.DAT',
     +     TYPE='OLD',READONLY)
        READ(2,1)XE(1),XE(3),XE(5)
        READ(2,1)XE(2),XE(4),XE(6)
        READ(2,1)XP(1),XP(3),XP(5)
        READ(2,1)XP(2),XP(4),XP(6)
        READ(2,6)TGO
        READ(2,8)A
        READ(2,8)MDOT
      CLOSE(2)
      PRINT *,' '
      PRINT *,' ENTER TIME STEP VALUE'
      READ *,H
      PRINT *,' '
      PRINT *,' ENTER CONTROL/CONSTRAINT MULTIPLIER'
      READ *,K
      PRINT *,' '
      PRINT *,' ENTER CONTROL EFFECTIVENESS RATIO'
      READ *,RHO
      IF (RHO .LT. 1.0) RHO=1.0

C  READ IN FILTER MEASUREMENT STANDARD DEVIATIONS
C  AND ASSIGN COVARIANCES TO R MATRIX DIAGONAL
7     FORMAT(F14.10)
      OPEN(UNIT=3,NAME='[ENGR.THESIS.SALFANO]
     +     FILTER8.REL',TYPE='OLD',READONLY)
      READ(3,7)SIGMAM(1)
      READ(3,7)SIGMAM(2)
      READ(3,7)SIGMAM(3)
      CLOSE(3)
      XR(1)=XE(1)-XP(1)
      XR(3)=XE(3)-XP(3)
      XR(5)=XE(5)-XP(5)
      RANGE=SQRT(XR(1)*XR(1)+XR(3)*XR(3)+XR(5)*XR(5))
      R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
      R3(2,2)=SIGMAM(2)*SIGMAM(2)
      R3(3,3)=SIGMAM(3)*SIGMAM(3)

C  READ IN THE NEXT SEED
      OPEN(UNIT=4,NAME='SIM.STATS',TYPE='OLD',READONLY)
9     FORMAT(2X,F9.5,2X,F9.3,2(2X,I2),2(2X,I5))
10    FORMAT(2X,I3,2X,I14)
      READ(4,10)SNUM,SEED
```

```
      PRINT *,' '
      PRINT *,' WHAT FILTER DO YOU CHOOSE ?'
      PRINT *,'   8  - EIGHT STATE EKF '
      PRINT *,'   6  -   SIX STATE EKF '
      PRINT *,'   60 -   SIX STATE EKF WITHOUT GRAVITY '
      READ *,FILTER

C  ZERO OUT OFF DIAGONAL FILTER MATRIX COMPONENTS
      DO 40 I=2,8
        JUP=I-1
        DO 40 J=1,JUP
          COV(I,J)=0.0
          COV(J,I)=0.0
          Q(I,J)=0.0
          Q(J,I)=0.0
40    CONTINUE

C  ESTABLISH ACCELERATION AND
C  MDOT PROPAGATION VARIANCES
      DUM=.1*MDOT*H
      Q(8,8)=DUM*DUM/H
      Q(7,7)=A*A*DUM*DUM*H

C  COMPUTE AND INITIALIZE PROPAGATION VARIANCES
C     COMPUTE TRANSFORMATION MATRIX
      CALL COMPTV(XP,TMAT)

C     TRANSFORM ESTIMATED STATE VARIABLES
      CALL TRANSFWD(XP(1),XP(3),XP(5),
     +        TMAT,XPT(1),XPT(3),XPT(5))
      CALL TRANSFWD(XP(2),XP(4),XP(6),
     +        TMAT,XPT(2),XPT(4),XPT(6))
      CALL TRANSFWD(XE(1),XE(3),XE(5),
     +        TMAT,XET(1),XET(3),XET(5))
      CALL TRANSFWD(XE(2),XE(4),XE(6),
     +        TMAT,XET(2),XET(4),XET(6))

C     INITIALIZE STATE VECTORS IN NEW FRAME
      DO 50 I=1,6
        XP(I)=XPT(I)
        XE(I)=XET(I)
        XED(I)=XET(I)
50    CONTINUE

C     ESTABLISH DUMMY TIME STEP
      DH=H/256

C     PROPAGATE DUMMY VARIABLES FORWARD ONE STEP
      TD=0.0
      AD=A
      MDOTD=MDOT
      DO 60 I=1,256
        CALL RK4SYSE(TD,XED,DH,AD,MDOTD)
        TD=TD+DH
60    CONTINUE
```

```
C       PROPAGATE TRANSFORMED VARIABLES FORWARD ONE STEP
        T=0.0
        IF (FILTER .EQ. 8) THEN
          AT=A
          MDOTT=MDOT
          CALL RK4SYSE(T,XET,H,AT,MDOTT)
        ENDIF
        IF (FILTER .EQ. 6) THEN
          AT=A+SQRT(H*Q(7,7))
          MDOTT=MDOT+SQRT(H*Q(8,8))
          CALL RK4SYSE(T,XET,H,AT,MDOTT)
        ENDIF
        IF (FILTER .EQ. 60) THEN
          XET(1)=XET(1)+H*XET(2)
          XET(3)=XET(3)+H*XET(4)
          XET(5)=XET(5)+H*XET(6)
          AT=A+SQRT(H*Q(7,7))
          DUM=1.0+H*AT/SQRT(XET(2)**2+XET(4)**2+
     +                      XET(6)**2)
          XET(2)=XET(2)*DUM
          XET(4)=XET(4)*DUM
          XET(6)=XET(6)*DUM
        ENDIF

C   COMPUTE REMAINING Q DIAGONAL COMPONENTS
          Q(1,1)=((XET(1)-XED(1))**2+(XET(3)-XED(3))**2+
     +           (XET(5)-XED(5))**2)/3.0/H
          Q(3,3)=Q(1,1)
          Q(5,5)=Q(1,1)
          Q(2,2)=((XET(2)-XED(2))**2+(XET(4)-XED(4))**2+
     +           (XET(6)-XED(6))**2)/3.0/H
          Q(4,4)=Q(2,2)
          Q(6,6)=Q(2,2)
          Q(7,7)=(AT-AD)*(AT-AD)/H

C   ASSIGN STARTUP COVARIANCES
          COV(1,1)=100.0
          COV(2,2)=SQRT(COV(1,1))
          COV(3,3)=COV(1,1)
          COV(4,4)=COV(2,2)
          COV(5,5)=COV(1,1)
          COV(6,6)=COV(2,2)
          COV(7,7)=(0.1*A)**2
          COV(8,8)=(0.1*MDOT)**2

C   INITIALIZE VARIABLES
          TOL=0.0001
          TSTART=3.0
          SFLAG=0
          KFLAG=0
          SW=0
          DELTAY=0.0
          DELTAZ=0.0
          VTOT=0.0
```

```
      MAXG=6.0
      MING=0.05*MAXG
      MAXDV=MAXG*10.0*H
      MINDV=MING*10.0*H
      COUNT=100

C  ASK USER TO CHOOSE CONTROL METHOD
      PRINT *,' '
      PRINT *,' WHAT CONTROL METHOD DO YOU CHOOSE ?'
      PRINT *,'   1 - PLAN A'
      PRINT *,'   2 - PLAN B'
      PRINT *,'   3 - PLAN C'
      PRINT *,'   4 - DUAL CONTROL'
      PRINT *,'   5 - CERTAINTY CONTROL'
      PRINT *,'   6 - TRUTH MODEL'
      READ *,PLAN
      UPDATE=1
      IF (PLAN .EQ. 4) UPDATE=2
      IF (PLAN .EQ. 5) UPDATE=0

C  ASK USER FOR NOISE OPTION
      PRINT *,' '
      PRINT *,' CHOOSE YOUR OPTION'
      PRINT *,'   1 - NO NOISE'
      PRINT *,'   2 - NOISE'
      PRINT *,'   3 - NOISE + SCREEN PRINTOUT'
      PRINT *,'   4 - NOISE + DATAFILE PRINTOUT'
      READ *,OPT

C  INITIALIZE ESTIMATED VARIABLES
      DO 100 I=1,6
        XPEST(I)=XP(I)
        IF (OPT .EQ. 1) THEN
          XEEST(I)=XE(I)
         ELSE
          XEEST(I)=XE(I)+SQRT(COV(I,I))*
     +                    GAUSS(SEED,SW,G2)
        ENDIF
        XHAT(I)=XEEST(I)-XPEST(I)
100   CONTINUE
      IF (OPT .EQ. 1) THEN
        XHAT(7)=A
        XHAT(8)=MDOT
       ELSE
        XHAT(7)=A+SQRT(COV(7,7))*GAUSS(SEED,SW,G2)
        XHAT(8)=MDOT+SQRT(COV(8,8))*GAUSS(SEED,SW,G2)
      ENDIF

C  OPEN UNITS FOR WRITING OUTPUT DATA
5     FORMAT(F13.7,2X,F13.7)
      IF (OPT .EQ. 4) THEN
        OPEN(UNIT=11,FILE='RES1.DAT',STATUS='NEW',
     +       IOSTAT=ISTAT)
```

```
      OPEN(UNIT=12,FILE='RES2.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=13,FILE='RES3.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=14,FILE='RES4.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=15,FILE='RES5.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=16,FILE='RES6.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=17,FILE='RES7.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=18,FILE='RES8.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=19,FILE='DELTAY.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=20,FILE='DELTAZ.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=21,FILE='COV1.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=22,FILE='COV2.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=23,FILE='COV3.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=24,FILE='COV4.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=25,FILE='COV5.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=26,FILE='COV6.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=27,FILE='COV7.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=28,FILE='COV8.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=29,FILE='MISS.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=30,FILE='TOL.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=31,FILE='COV1M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=32,FILE='COV2M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=33,FILE='COV3M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=34,FILE='COV4M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=35,FILE='COV5M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=36,FILE='COV6M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=37,FILE='COV7M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      OPEN(UNIT=38,FILE='COV8M.DAT',STATUS='NEW',
     +     IOSTAT=ISTAT)
      ENDIF
```

```
C  *************************
C  * BEGIN SIMULATION LOOP *
C  *************************


      DO 990 SIMCNT=1,50000

C  PRINT RESIDUALS, VELOCITY CHANGES,
C  AND COVARIANCES TO DATAFILES
        IF (OPT .EQ. 4) THEN
          WRITE(11,5),T,XEEST(1)-XE(1)
          WRITE(12,5),T,XEEST(2)-XE(2)
          WRITE(13,5),T,XEEST(3)-XE(3)
          WRITE(14,5),T,XEEST(4)-XE(4)
          WRITE(15,5),T,XEEST(5)-XE(5)
          WRITE(16,5),T,XEEST(6)-XE(6)
          WRITE(17,5),T,XHAT(7)-A
          WRITE(18,5),T,XHAT(8)-MDOT
          IF ((DELTAY .NE. 0.0) .OR. (SIMCNT .EQ. 1))
     +       WRITE(19,5),T,DELTAY
          IF ((DELTAZ .NE. 0.0) .OR. (SIMCNT .EQ. 1))
     +       WRITE(20,5),T,DELTAZ
          WRITE(21,5),T,SQRT(COV(1,1))
          WRITE(22,5),T,SQRT(COV(2,2))
          WRITE(23,5),T,SQRT(COV(3,3))
          WRITE(24,5),T,SQRT(COV(4,4))
          WRITE(25,5),T,SQRT(COV(5,5))
          WRITE(26,5),T,SQRT(COV(6,6))
          WRITE(27,5),T,SQRT(COV(7,7))
          WRITE(28,5),T,SQRT(COV(8,8))
          WRITE(29,5),T,SQRT(MISS2)
          IF (KDEVF .GT. 0.0) WRITE(30,5),T,KDEVF
          WRITE(31,5),T,-SQRT(COV(1,1))
          WRITE(32,5),T,-SQRT(COV(2,2))
          WRITE(33,5),T,-SQRT(COV(3,3))
          WRITE(34,5),T,-SQRT(COV(4,4))
          WRITE(35,5),T,-SQRT(COV(5,5))
          WRITE(36,5),T,-SQRT(COV(6,6))
          WRITE(37,5),T,-SQRT(COV(7,7))
          WRITE(38,5),T,-SQRT(COV(8,8))
        ENDIF

C  TEST TO SEE IF TIME IS UP
        IF (TGO .LE. H) GOTO 995

C  REASSIGN DUMMY VARIABLES
        DO 105 I=1,6
          XPD(I)=XPEST(I)
          XED(I)=XEEST(I)
105     CONTINUE
```

```
      IF ((UPDATE .NE. 1) .AND. (T .GE. TSTART)) THEN
        DO 106 I=1,8
          XDUAL(I)=XHAT(I)
          DO 106 J=1,8
            CVDUAL(I,J)=COV(I,J)
106     CONTINUE
      ENDIF

C  PROPAGATE DUMMY VARIABLES FORWARD ONE STEP
      TD=T
      IF ((UPDATE .NE. 1) .AND. (T .GE. TSTART)) THEN
        RANGE=SQRT(XDUAL(1)*XDUAL(1)+XDUAL(3)*XDUAL(3)+
     +           XDUAL(5)*XDUAL(5))
        R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
        IF (FILTER .EQ. 8) CALL EKF8(XDUAL,XED,XPD,TD,
     +     H,CVDUAL,Q,R3,0.0,0.0,0.0,KFLAG,RES,UPDATE)
        IF (FILTER .EQ. 6) THEN
          Q(2,2)=1.0+XDUAL(7)*H*Q(8,8)
          Q(4,4)=Q(2,2)
          Q(6,6)=Q(2,2)
          CALL EKF6(XDUAL,XED,XPD,TD,H,CVDUAL,Q,
     +          R3,0.0,0.0,0.0,KFLAG,RES,UPDATE)
        ENDIF
        AD=XDUAL(7)
        MDOTD=XDUAL(8)
       ELSE
        AD=XHAT(7)
        MDOTD=XHAT(8)
        CALL RK4SYSP(TD,XPD,H)
        CALL RK4SYSE(TD,XED,H,AD,MDOTD)
      ENDIF
      TD=TD+H
      TGO=TGO-H

C  INITIALIZE TRANSFORMED VARIABLES TO DUMMY VARIABLES
      AT=AD
      MDOTT=MDOTD
      DO 110 I=1,6
        XET(I)=XED(I)
        XPT(I)=XPD(I)
110   CONTINUE
      IF (PLAN .EQ. 4) THEN
        DO 111 I=1,8
          XDUALD(I)=XDUAL(I)
          DO 111 J=1,8
            CVDD(I,J)=CVDUAL(I,J)
111     CONTINUE
        DO 112 I=1,6
          XEDD(I)=XED(I)
          XPDD(I)=XPD(I)
112     CONTINUE
      ENDIF
```

```
C  STORE INITIAL VALUES OF DEVIATIONS FOR
C  SPLINE COMPUTATION
       IF ((PLAN .EQ. 5) .AND.
     +     (T .GE. TSTART)) THEN
         SIGT0(1)=SQRT(CVDUAL(1,1))
         SIGT0(2)=(SIGT0(1)-SQRT(COV(1,1)))/H
         SIGT0(3)=SQRT(CVDUAL(3,3))
         SIGT0(4)=(SIGT0(3)-SQRT(COV(3,3)))/H
         SIGT0(5)=SQRT(CVDUAL(5,5))
         SIGT0(6)=(SIGT0(5)-SQRT(COV(5,5)))/H
       ENDIF

C  ESTABLISH SPLINE TIME STEP
       IF (TGO/COUNT .LT. H) COUNT=COUNT-1
       IF (COUNT .LT. 1) COUNT=1
       DH=TGO/COUNT

C  PROPAGATE DUMMY VARIABLES FORWARD
C  TO PREDICTED IMPACT TIME
       IF ((PLAN .EQ. 5) .AND.
     +     (T .GE. TSTART)) THEN
         DO 113 I=1,COUNT
           IF (I .EQ. COUNT) THEN
             CVD(1)=CVDUAL(1,1)
             CVD(3)=CVDUAL(3,3)
             CVD(5)=CVDUAL(5,5)
           ENDIF
           RANGE=SQRT(XDUAL(1)*XDUAL(1)+XDUAL(3)*
     +                XDUAL(3)+XDUAL(5)*XDUAL(5))
           R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
           IF (FILTER .EQ. 8) THEN
             CALL EKF8(XDUAL,XED,XPD,TD,DH,CVDUAL,Q,R3,
     +                 0.0,0.0,0.0,KFLAG,RES,UPDATE)
           ENDIF
           IF (FILTER .EQ. 6) THEN
             Q(2,2)=1.0+XDUAL(7)*DH*Q(8,8)
             Q(4,4)=Q(2,2)
             Q(6,6)=Q(2,2)
             CALL EKF6(XDUAL,XED,XPD,TD,DH,CVDUAL,Q,
     +                 R3,0.0,0.0,0.0,KFLAG,RES,UPDATE)
           ENDIF
           TD=TD+DH
113      CONTINUE
         SIGTF(1)=SQRT(CVDUAL(1,1))
         SIGTF(2)=(SIGTF(1)-SQRT(CVD(1)))/DH
         SIGTF(3)=SQRT(CVDUAL(3,3))
         SIGTF(4)=(SIGTF(3)-SQRT(CVD(3)))/DH
         SIGTF(5)=SQRT(CVDUAL(5,5))
         SIGTF(6)=(SIGTF(5)-SQRT(CVD(5)))/DH
         KDEVF=SQRT(K*(CVDUAL(1,1)+CVDUAL(3,3)+
     +                 CVDUAL(5,5)))
       ELSE
```

```
            DO 115 I=1,COUNT
              CALL RK4SYSP(TD,XPD,DH)
              CALL RK4SYSE(TD,XED,DH,AD,MDOTD)
              TD=TD+DH
115           CONTINUE
            ENDIF

C   COMPUTE FINAL ESTIMATED RELATIVE STATES
            DO 120 I=1,6
              XR(I)=XED(I)-XPD(I)
120         CONTINUE

C   COMPUTE ESTIMATED MISS DISTANCE SQUARED
            MISS2=XR(1)*XR(1)+XR(3)*XR(3)+XR(5)*XR(5)

C   UPDATE DUMMY TIME-TO-GO
            DTGO=TD-T-H

C   COMPUTE VELOCITY CHANGES
            IF ((TGO .GT. H) .AND.
     +          (T .GE. TSTART)) THEN

C             COMPUTE RELATIVE SPLINE COEFFICIENTS
              CALL SPLINE(XET(1)-XPT(1),XET(2)-XPT(2),XR(1),
     +             XR(2),DTGO,AX,BX,CX,DX)
              CALL SPLINE(XET(3)-XPT(3),XET(4)-XPT(4),XR(3),
     +             XR(4),DTGO,AY,BY,CY,DY)
              CALL SPLINE(XET(5)-XPT(5),XET(6)-XPT(6),XR(5),
     +             XR(6),DTGO,AZ,BZ,CZ,DZ)

              IF (PLAN .EQ. 5) THEN
                CALL SPLINE(SIGT0(1),SIGT0(2),SIGTF(1),
     +               SIGTF(2),DTGO,ASIGX,BSIGX,CSIGX,DSIGX)
                CALL SPLINE(SIGT0(3),SIGT0(4),SIGTF(3),
     +               SIGTF(4),DTGO,ASIGY,BSIGY,CSIGY,DSIGY)
                CALL SPLINE(SIGT0(5),SIGT0(6),SIGTF(5),
     +               SIGTF(6),DTGO,ASIGZ,BSIGZ,CSIGZ,DSIGZ)
              ENDIF

C             COMPUTE OPTIMAL CHANGES IN VELOCITY AND
C             IMPACT TIME
              IF (PLAN .EQ. 1) THEN
                CALL SEARCHA(AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,
     +               CZ,DZ,K,TGO,DELTAY,DELTAZ,TOL,SFLAG)
              ENDIF

              IF (PLAN .EQ. 2) THEN
                CALL SEARCHB(AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,
     +               CZ,DZ,TGO,DELTAY,DELTAZ,TOL,SFLAG)
              ENDIF
```

```
      IF (PLAN .EQ. 3) THEN
        DO 135 I=1,6
          XR(I)=XET(I)-XPT(I)
135     CONTINUE
        CALL SEARCHC(XR,XET(2),XET(4),XET(6),AT,
     +             MDOT,TGO,DELTAY,DELTAZ,TOL,SFLAG)
      ENDIF

      IF (PLAN .EQ. 4) THEN
        CALL SEARCHD(AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,
     +             CZ,DZ,K,TGO,DELTAY,DELTAZ,TOL,SFLAG,
     +             XDDALD,XEDD,XPDD,CVDD,MAXDV,COUNT,
     +             Q,R3,SIGMAM,H)
      ENDIF

      IF (PLAN .EQ. 5) THEN
        CALL SEARCHCC(AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,
     +       CZ,DZ,ASIGX,BSIGX,CSIGX,DSIGX,ASIGY,BSIGY,
     +       CSIGY,DSIGY,ASIGZ,BSIGZ,CSIGZ,DSIGZ,K,TGO,
     +       DELTAY,DELTAZ,TOL,SFLAG)
      ENDIF

      IF (PLAN .EQ. 6) THEN
        CALL SEARCHT(XPT,XET,AT,MDOTT,H,T,TGO,
     +                   DELTAY,DELTAZ,TOL,SFLAG,COUNT)
      ENDIF

C     BOUND VELOCITY CHANGES
      IF (ABS(DELTAY) .LT. MINDV) THEN
        DELTAY=0.0
       ELSE
        IF (DELTAY .GT. MAXDV) DELTAY=MAXDV
        IF (DELTAY .LT. -MAXDV) DELTAY=-MAXDV
      ENDIF
      IF (ABS(DELTAZ) .LT. MINDV) THEN
        DELTAZ=0.0
       ELSE
        IF (DELTAZ .GT. MAXDV) DELTAZ=MAXDV
        IF (DELTAZ .LT. -MAXDV) DELTAZ=-MAXDV
      ENDIF

      DV=ABS(DELTAY)+ABS(DELTAZ)
      IF (DV .GE. MAXDV) THEN
        TSTART=T
       ELSE
        TSTART=T+(RHO-1.0)*TGO/RHO
      ENDIF

     ELSE

      DELTAY=0.0
      DELTAZ=0.0
      DV=0.0
```

```
          ENDIF

          IF (DV .GE. MINDV) WRITE(*,1)DELTAY,DELTAZ,TGO

C   PROPAGATE REAL VARIABLES AND PURSUER ESTIMATE
C   FORWARD ONE STEP (I.M.U. ASSUMED PERFECT)
          CALL RK4SYSP(T,XP,H)
          CALL RK4SYSE(T,XE,H,A,MDOT)
          DO 210 I=1,6
            XPEST(I)=XP(I)
210       CONTINUE

C   COMPUTE SENSOR MEASUREMENTS (PLUS NOISE)
          XR(1)=XE(1)-XP(1)
          XR(3)=XE(3)-XP(3)
          XR(5)=XE(5)-XP(5)
          RANGE=SQRT(XR(1)*XR(1)+XR(3)*XR(3)+XR(5)*XR(5))
          R3(1,1)=RANGE*SIGMAM(1)*RANGE*SIGMAM(1)
          THETA=ATAN(XR(5)/XR(1))
          GAMMA=ATAN(XR(3)/XR(1))
          IF (OPT .NE. 1) THEN
            RANGE=RANGE*(1.0+GAUSS(SEED,SW,G2)*SIGMAM(1))
            THETA=THETA+GAUSS(SEED,SW,G2)*SIGMAM(2)
            GAMMA=GAMMA+GAUSS(SEED,SW,G2)*SIGMAM(3)
          ENDIF

C   GET FILTER ESTIMATES
          IF (FILTER .EQ. 8) CALL EKF8(XHAT,XEEST,XPEST,
     +      T,H,COV,Q,R3,RANGE,THETA,GAMMA,KFLAG,RES,1)
          IF (FILTER .EQ. 6) THEN
            Q(2,2)=1.0+XHAT(7)*H*Q(8,8)
            Q(4,4)=Q(2,2)
            Q(6,6)=Q(2,2)
            CALL EKF6(XHAT,XEEST,XPEST,T,H,
     +           COV,Q,R3,RANGE,THETA,GAMMA,KFLAG,RES,1)
          ENDIF

          IF (FILTER .EQ. 60) CALL EKF60(XHAT,XEEST,
     +    XPEST,T,H,COV,Q,R3,RANGE,THETA,GAMMA,KFLAG,RES)

C   UPDATE EVADER ESTIMATE USING RELATIVE ESTIMATE
          DO 200 I=1,6
            XEEST(I)=XPEST(I)+XHAT(I)
200       CONTINUE

C   RECURSIVELY COMPUTE MEAN AND VARIANCE OF
C   MEASUREMENT RESIDUALS
          IF (SIMCNT .EQ. 1) THEN
            VAR(1)=RES(1)*RES(1)
            VAR(2)=RES(2)*RES(2)
            VAR(3)=RES(3)*RES(3)
           ELSE
            VAR(1)=VAR(1)*(SIMCNT-2)/(SIMCNT-1)+
     +           (MEAN(1)-RES(1))*(MEAN(1)-RES(1))/SIMCNT
```

```
            VAR(2)=VAR(2)*(SIMCNT-2)/(SIMCNT-1)+
     +           (MEAN(2)-RES(2))*(MEAN(2)-RES(2))/SIMCNT
            VAR(3)=VAR(3)*(SIMCNT-2)/(SIMCNT-1)+
     +           (MEAN(3)-RES(3))*(MEAN(3)-RES(3))/SIMCNT
          ENDIF
          MEAN(1)=(MEAN(1)*(SIMCNT-1)+RES(1))/SIMCNT
          MEAN(2)=(MEAN(2)*(SIMCNT-1)+RES(2))/SIMCNT
          MEAN(3)=(MEAN(3)*(SIMCNT-1)+RES(3))/SIMCNT

C   PRINT ESTIMATED AND TRUE STATES AND COVARIANCES
          IF (OPT .EQ. 3) THEN
            PRINT *,' '
            PRINT *,' ESTIMATED STATE, TRUE STATE, ERROR'
            PRINT *,' AND COVARIANCE'
            DO 220 I=1,6
              PRINT *,XHAT(I),XE(I)-XP(I),
     +                XHAT(I)-XE(I)+XP(I)
              PRINT *,'    ',COV(I,I)
220         CONTINUE
          ENDIF

C   APPLY VELOCITY CHANGES
          XP(4)=XP(4)+DELTAY
          XP(6)=XP(6)+DELTAZ
          XPEST(4)=XP(4)
          XPEST(6)=XP(6)
          XHAT(4)=XHAT(4)-DELTAY
          XHAT(6)=XHAT(6)-DELTAZ
          VTOT=VTOT+DV

C   UPDATE TIME
          T=T+H

990     CONTINUE


C   ************************
C   * END SIMULATION LOOP *
C   ************************


995     CONTINUE

C   PRINT SQUARE ROOT OF COVARIANCE DIAGONAL
        PRINT *,' '
        PRINT *,' DEVIATIONS, ERROR'
        DO 800 I=1,6
          PRINT *,SQRT(COV(I,I)),XEEST(I)-XE(I)
800     CONTINUE
        PRINT *,SQRT(COV(7,7)),XHAT(7)-A
        PRINT *,SQRT(COV(8,8)),XHAT(8)-MDOT
```

```
C  PROPAGATE REAL DATA TO FINAL PREDICTED IMPACT TIME
      DH=TGO
      CALL RK4SYSP(T,XP,DH)
      CALL RK4SYSE(T,XE,DH,A,MDOT)
      T=T+DH

C  ITERATE TO FIND POINT OF CLOSEST APPROACH
      DH=H
      DO 310 J=1,25
        DO 300 I=1,6
          XR(I)=XE(I)-XP(I)
300     CONTINUE
        IF (ABS(DH) .LE. 0.0000001) GOTO 320
        DH=XR(1)*XR(2)+XR(3)*XR(4)+XR(5)*XR(6)
        DH=-DH/(XR(2)*XR(2)+XR(4)*XR(4)+XR(6)*XR(6))
        CALL RK4SYSP(T,XP,DH)
        CALL RK4SYSE(T,XE,DH,A,MDOT)
        T=T+DH
310   CONTINUE

320   CONTINUE

C  PRINT CONVERGENGE MESSAGE
      PRINT *,' '
      PRINT *,' SEARCH NON-CONVERGENCE =',SFLAG
      PRINT *,'   GAIN NON-CONVERGENCE =',KFLAG
      PRINT *,' '

C  PRINT TIME AND MISS OF CLOSEST APPROACH
      PRINT *,' IMPACT TIME : ',T
      RANGE=SQRT(XR(1)*XR(1)+XR(3)*XR(3)+XR(5)*XR(5))
      PRINT *,' TOTAL VELOCITY CHANGE : ',VTOT
      PRINT *,' MISS DISTANCE : ',RANGE
      PRINT *,XR(1),XR(3),XR(5)

C  CLOSE OUTPUT DATA FILES
      IF (OPT .EQ. 4) THEN
        DO 340 I=11,38
          CLOSE(I)
340     CONTINUE
      ENDIF

C  READ IN PREVIOUS SIMULATION DATA, SORT ON RANGE,
C  AND WRITE TO NEW FILE

      OPEN(UNIT=5,FILE='SIM.STATS',STATUS='NEW',
     +     IOSTAT=ISTAT)

      WRITE(5,10)(SNUM+1),SEED
      SRANGE=0.0
C      RANGE=SIGN(RANGE,XR(3))
      J=0
```

```
      DO 350 I=1,SNUM
        READ(4,9)SRANGE,SVTOT,SPLAN,SFILTR,SSFLAG,SKFLAG
        IF ((J .EQ. 0) .AND. (RANGE .LE. SRANGE))THEN
           WRITE(5,9)RANGE,VTOT,PLAN,FILTER,SFLAG,KFLAG
           J=1
        ENDIF
        WRITE(5,9)SRANGE,SVTOT,SPLAN,SFILTR,SSFLAG,SKFLAG
350   CONTINUE
      IF (J .EQ. 0)
     +      WRITE(5,9)RANGE,VTOT,PLAN,FILTER,SFLAG,KFLAG

      CLOSE(4)
      CLOSE(5)

      END
```

```
C     TOOL1

C  THIS IS A COLLECTION OF SUBROUTINES NEEDED FOR
C  ORBIT PROPAGATION
C  IN THE HYPERVELOCITY ORBITAL INTERCEPT PROGRAM

C  SUBROUTINE DICTIONARY

C     A          EVADER ACCELERATION DUE TO THRUSTING
C     AD         DUMMY ACCELERATION
C     COUNT      ITERATION FINAL COUNT
C     MDOT       UNITIZED MASS FLOW RATE OF EVADER
C     MDOTD      DUMMY MASS FLOW RATE OF EVADER
C     H          STEP SIZE
C     I          ITERATION COUNTER
C     T          TIME
C     TFINAL     FINAL TIME
C     XE         STATE VECTOR OF EVADER
C     XP         STATE VECTOR OF PURSUER



      SUBROUTINE XPSYSP(X,F)

C  THIS SUBROUTINE EVALUATES THE FUNCTIONS FOR RK4SYSP
C  (ORBITAL DYNAMICS FOR TWO BODY PROBLEM FOR PURSUER)
C  (X VECTOR = [X XDOT Y YDOT Z ZDOT])
C  (F VECTOR = [XDOT XDOUBLEDOT YDOT YDOUBLEDOT
C               ZDOT ZDOUBLEDOT])

      REAL*8 X(6),F(6),RSQRD,CONST

      RSQRD=X(1)*X(1)+X(3)*X(3)+X(5)*X(5)
      CONST=-3.986012E14/RSQRD/SQRT(RSQRD)
      F(1)=X(2)
      F(2)=CONST*X(1)
      F(3)=X(4)
      F(4)=CONST*X(3)
      F(5)=X(6)
      F(6)=CONST*X(5)

      END



      SUBROUTINE RK4SYSP(T,X,H)

C  THIS SUBROUTINE IS A ONE STEP RUNGE-KUTTA
C  4TH ORDER INTEGRATOR FOR THE PURSUER DYNAMICS
```

```
      REAL*8 T,X(6),H,F1(6),F2(6),F3(6)
      REAL*8 F4(6),H2,DUMX(6)
      INTEGER I

      H2=0.5*H

C  FIND F1
      CALL XPSYSP(X,F1)

C  FIND F2
      DO 100 I=1,6
        DUMX(I)=X(I)+H2*F1(I)
100   CONTINUE
      CALL XPSYSP(DUMX,F2)

C  FIND F3
      DO 200 I=1,6
        DUMX(I)=X(I)+H2*F2(I)
200   CONTINUE
      CALL XPSYSP(DUMX,F3)

C  FIND F4
      DO 300 I=1,6
        DUMX(I)=X(I)+H*F3(I)
300   CONTINUE
      CALL XPSYSP(DUMX,F4)

C  UPDATE THE STATE
      DO 400 I=1,6
        X(I)=X(I)+H*(F1(I)+F2(I)+F2(I)+F3(I)+
     +               F3(I)+F4(I))/6.0
400   CONTINUE

      END



      SUBROUTINE EULERP(T,X,H)

C  THIS SUBROUTINE IS A ONE STEP EULER INTEGRATOR
C  FOR THE PURSUER DYNAMICS

      REAL*8 T,X(6),F1(6),H
      INTEGER I

C  FIND F1
      CALL XPSYSP(X,F1)

C  UPDATE THE STATE
      DO 400 I=1,6
        X(I)=X(I)+H*F1(I)
400   CONTINUE

      END
```

```
      SUBROUTINE XPSYSE(X,F,A,MDOT,T)

C  THIS SUBROUTINE EVALUATES THE FUNCTIONS FOR RK4SYSE
C  (ORBITAL DYNAMICS FOR TWO BODY PROBLEM FOR EVADER
C   USING THE ROCKET EQUATION:
C         A=Ao/(1-MDOT*T)
C  (X VECTOR = [X XDOT Y YDOT Z ZDOT])
C  (F VECTOR = [XDOT XDOUBLEDOT YDOT YDOUBLEDOT ZDOT
C                ZDOUBLEDOT ADOT])

      REAL*8 X(6),F(8),RSQRD,CONST,MDOT,T,A,V

      RSQRD=X(1)*X(1)+X(3)*X(3)+X(5)*X(5)
      CONST=-3.986012E14/RSQRD/SQRT(RSQRD)
      V=SQRT(X(2)*X(2)+X(4)*X(4)+X(6)*X(6))

      F(1)=X(2)
      F(2)=CONST*X(1)+A*X(2)/V
      F(3)=X(4)
      F(4)=CONST*X(3)+A*X(4)/V
      F(5)=X(6)
      F(6)=CONST*X(5)+A*X(6)/V
      F(7)=A*MDOT
      F(8)=MDOT*MDOT

      END



      SUBROUTINE RK4SYSE(T,X,H,A,MDOT)

C  THIS SUBROUTINE IS A ONE STEP RUNGE-KUTTA
C  4TH ORDER INTEGRATOR FOR THE EVADER DYNAMICS

      REAL*8 T,X(6),H,F1(8),F2(8),F3(8),F4(8)
      REAL*8 H2,DUMX(6),A,MDOT,AD,MDOTD
      INTEGER I

      H2=0.5*H

C  FIND F1
      CALL XPSYSE(X,F1,A,MDOT,T)

C  FIND F2
      DO 100 I=1,6
        DUMX(I)=X(I)+H2*F1(I)
100   CONTINUE
      AD=A+H2*F1(7)
      MDOTD=MDOT+H2*F1(8)
      CALL XPSYSE(DUMX,F2,AD,MDOTD,T)
```

```
C  FIND F3
      DO 200 I=1,6
        DUMX(I)=X(I)+H2*F2(I)
200   CONTINUE
      AD=A+H2*F2(7)
      MDOTD=MDOT+H2*F2(8)
      CALL XPSYSE(DUMX,F3,AD,MDOTD,T)

C  FIND F4
      DO 300 I=1,6
        DUMX(I)=X(I)+H*F3(I)
300   CONTINUE
      AD=A+H*F3(7)
      MDOTD=MDOT+H2*F3(8)
      CALL XPSYSE(DUMX,F4,AD,MDOTD,T)

C  UPDATE THE STATE
      DO 400 I=1,6
        X(I)=X(I)+H*(F1(I)+F2(I)+F2(I)+F3(I)+
     +               F3(I)+F4(I))/6.0
400   CONTINUE
      A=A+H*(F1(7)+F2(7)+F2(7)+F3(7)+
     +       F3(7)+F4(7))/6.0
      MDOT=MDOT+H*(F1(8)+F2(8)+F2(8)+F3(8)+
     +             F3(8)+F4(8))/6.0

      END



      SUBROUTINE EULERE(T,X,H,A,MDOT)

C  THIS SUBROUTINE IS A ONE STEP EULER INTEGRATOR
C  FOR THE EVADER DYNAMICS

      REAL*8 T,X(6),F1(8),H,A,MDOT
      INTEGER I

C  FIND F1
      CALL XPSYSE(X,F1,A,MDOT,T)

C  UPDATE THE STATE
      DO 400 I=1,6
        X(I)=X(I)+H*F1(I)
400   CONTINUE
      A=A+H*F1(7)
      MDOT=MDOT+H*F1(8)

      END
```

```
C     TOOL2

C  THIS IS A COLLECTION OF SUBROUTINES NEEDED FOR
C  COORDINATE TRANSFORMATIONS
C  IN THE HYPERVELOCITY ORBITAL INTERCEPT PROGRAM

C  SUBROUTINE DICTIONARY

C     XHAT       ESTIMATED RELATIVE STATE VECTOR
C     XP         STATE VECTOR OF PURSUER
C     XPD        DUMMY STATE VECTOR OF PURSUER
C     TMAT       TRANSFORMATION MATRIX
C     TMATA      TRANSFORMATION MATRIX



      SUBROUTINE COMPTLOS(XHAT,TMAT)

C  THIS SUBROUTINE COMPUTES THE MATRIX (TMAT) THAT
C  TRANSFORMS THE REFERENCE FRAME TO THE LOS FRAME
C  WHERE THE X AXIS OF THE LOS FRAME LIES ALONG THE
C  RELATIVE POSTION VECTOR.

      REAL*8 XHAT(6),TMAT(3,3),A,R,AR

      R=SQRT(XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)+
     +       XHAT(5)*XHAT(5))
      A=SQRT(XHAT(1)*XHAT(1)+XHAT(5)*XHAT(5))
      IF (A .LT. .00001) THEN A=.00001
      AR=A*R
      TMAT(1,1)=XHAT(1)/R
      TMAT(1,2)=XHAT(3)/R
      TMAT(1,3)=XHAT(5)/R
      TMAT(2,1)=-XHAT(1)*XHAT(3)/AR
      TMAT(2,2)=A/R
      TMAT(2,3)=-XHAT(3)*XHAT(5)/AR
      TMAT(3,1)=-XHAT(5)/A
      TMAT(3,2)=0.0
      TMAT(3,3)=XHAT(1)/A

      END



      SUBROUTINE COMPTV(XP,TMAT)

C  THIS SUBROUTINE COMPUTES THE MATRIX (TMAT) THAT
C  TRANSFORMS THE REFERENCE FRAME TO THE BODY FRAME
C  WHERE THE X AXIS OF THE BODY FRAME LIES ALONG
C  THE PURSUER'S VELOCITY VECTOR AND THE PURSUER'S
C  RADIUS VECTOR IS IN THE NEW XY PLANE.
```

```
      REAL*8 XP(6),XPD(6),TMATA(3,3),TMAT(3,3)
      REAL*8 A,B,V,AV

      V=SQRT(XP(2)*XP(2)+XP(4)*XP(4)+XP(6)*XP(6))
      A=SQRT(XP(2)*XP(2)+XP(6)*XP(6))
      IF (A .LT. .00001) THEN A=.00001
      AV=A*V
      TMAT(1,1)=XP(2)/V
      TMAT(1,2)=XP(4)/V
      TMAT(1,3)=XP(6)/V
      TMAT(2,1)=-XP(2)*XP(4)/AV
      TMAT(2,2)=A/V
      TMAT(2,3)=-XP(4)*XP(6)/AV
      TMAT(3,1)=-XP(6)/A
      TMAT(3,2)=0.0
      TMAT(3,3)=XP(2)/A

      CALL TRANSFWD(XP(1),XP(3),XP(5),TMAT,XPD(1),
     +              XPD(3),XPD(5))
      B=SQRT(XPD(3)*XPD(3)+XPD(5)*XPD(5))
      IF (B .LT. .0001) RETURN
      TMATA(2,1)=(TMAT(2,1)*XPD(3)+TMAT(3,1)*XPD(5))/B
      TMATA(3,1)=(TMAT(3,1)*XPD(3)-TMAT(2,1)*XPD(5))/B
      TMATA(2,2)=(TMAT(2,2)*XPD(3)+TMAT(3,2)*XPD(5))/B
      TMATA(3,2)=(TMAT(3,2)*XPD(3)-TMAT(2,2)*XPD(5))/B
      TMATA(2,3)=(TMAT(2,3)*XPD(3)+TMAT(3,3)*XPD(5))/B
      TMATA(3,3)=(TMAT(3,3)*XPD(3)-TMAT(2,3)*XPD(5))/B
      TMAT(2,1)=TMATA(2,1)
      TMAT(3,1)=TMATA(3,1)
      TMAT(2,2)=TMATA(2,2)
      TMAT(3,2)=TMATA(3,2)
      TMAT(2,3)=TMATA(2,3)
      TMAT(3,3)=TMATA(3,3)

      END



      SUBROUTINE TRANSFWD(X,Y,Z,TMAT,XT.YT,ZT)

C     THIS ROUTINE TAKES THE X,Y,Z VECTOR AND USES
C     THE TRANSFORMATION MATRIX TMAT TO FORM THE
C     VECTOR XT,YT,ZT

      REAL*8 X,Y,Z,TMAT(3,3),XT,YT,ZT
      INTEGER I

      XT=TMAT(1,1)*X+TMAT(1,2)*Y+TMAT(1,3)*Z
      YT=TMAT(2,1)*X+TMAT(2,2)*Y+TMAT(2,3)*Z
      ZT=TMAT(3,1)*X+TMAT(3,2)*Y+TMAT(3,3)*Z

      END
```

```
      SUBROUTINE TRANSBKWD(X,Y,Z,TMAT,XT,YT,ZT)

C  THIS ROUTINE TAKES THE X,Y,Z VECTOR AND USES THE
C  INVERSE OF THE TRANSFORMATION MATRIX TMAT TO
C  FORM THE VECTOR XT,YT,ZT

      REAL*8 X,Y,Z,TMAT(3,3),XT,YT,ZT
      INTEGER I

      XT=TMAT(1,1)*X+TMAT(2,1)*Y+TMAT(3,1)*Z
      YT=TMAT(1,2)*X+TMAT(2,2)*Y+TMAT(3,2)*Z
      ZT=TMAT(1,3)*X+TMAT(2,3)*Y+TMAT(3,3)*Z

      END
```

```
C     TOOL3

C  THIS IS A COLLECTION OF SUBROUTINES NEEDED FOR
C  EXTENDED KALMAN FILTERING
C  IN THE HYPERVELOCITY ORBITAL INTERCEPT PROGRAM

C  SUBROUTINE DICTIONARY

C      ACCDEN    DENOMINATOR OF EVADER ACCELERATION TERM
C      ACCEL     PRESENT EVADER THRUSTING ACCELERATION
C      COV       COVARIANCE MATRIX
C      DET       THE DETERMINANT OF THE MATRIX 'MAT'
C      DMAT      DUMMY MATRIX
C      DCMAT     DUMMY COLUMN MATRIX
C      DT        TIME STEP
C      DUM       DUMMY VARIABLE
C      E         RADIUS OF EVADER
C      E2        E**2
C      E3        E**3
C      E5        E**5
C      EDOT      VELOCITY OF EVADER
C      EDOT2     EDOT**2
C      EDOT3     EDOT**3
C      F         MATRIX OF STATE PARTIAL DERIVATIVES
C      GAIN      GAIN MATRIX
C      GAMMA     OBSERVED LINE-OF-SIGHT ANGLE (IN PLANE)
C      H         MATRIX OF MEASUREMENT PARTIALS
C      HYP13     XHAT(1)**2 + XHAT(3)**2
C      HYP15     XHAT(1)**2 + XHAT(5)**2
C      I         COUNTER
C      J         COUNTER
C      JUP       UPPER LIMIT ON 'J' COUNTER
C      K         COUNTER
C      KFLAG     GAIN CONVERGENCE FLAG
C      MEAN      GAUSSIAN MEAN
C                  1 - WITHOUT FILTER
C                  2 - WITH FILTER
C                  3 - WITH FILTER + PRINTOUT
C      PDOT      TIME DERIVATIVE OF COVARIANCE MATRIX
C      POLD      PROPAGATED COVARIANCE MATRIX
C      Q         VARIANCE OF FILTER STATE NOISE
C      R         ESTIMATE OF RANGE OF EVADER FROM PURSUER
C      RES       MEASUREMENT RESIDUALS
C      R3        VARIANCE OF MEASUREMENT NOISE
C      RANGE     MEASUREMENT OF RELATIVE RANGE
C      SEED      SEED FOR RANDOM NUMBER GENERATOR
C      SIGMA     GAUSSIAN STANDARD DEVIATION
C      SUM       SUM OF UNIFORMLY DISTRIBUTED NUMBERS
C      T         TIME
C      THETA     OBSERVED OUT-OF-PLANE LOS ANGLE
C      U         GRAVITATIONAL CONSTANT
```

```
C     UPDATE    FLAG FOR UPDATING EKF
C                0  NO UPDATE
C                1  UPDATE
C                2  UPDATE WITH RESIDUALS EQUAL TO ZERO
C     V         ESTIMATE OF RELATIVE RANGE RATE
C     VAR       VARIANCE OF MEASUREMENT
C     XEEST     ESITIMATED EVADER VECTOR
C     XHAT      ESTIMATE OF X VECTOR (BODY FRAME)
C                1  RELATIVE X POSITICN
C                2  RELATIVE X VELOCITY
C                3  RELATIVE Y POSITION
C                4  RELATIVE Y VELOCITY
C                5  RELATIVE Z POSITION
C                6  RELATIVE Z VELOCITY
C                7  PRESENT ACCELERATION OF EVADER
C                8  BOOSTER MASS FLOW RATE
C     XPEST     ESTIMATED PURSUER VECTOR



      FUNCTION GAUSS(SEED,SW,G2)
C  CREATES A POINT HAVING A GAUSSIAN DISTRIBUTION WITH
C     MEAN=0.0
C     SIGMA=1.0

      REAL MTH$RANDOM,A,B
      REAL*8 GAUSS,G2
      INTEGER SEED,SW,I

      IF (SW .GT. 0) THEN
        SW=0
        GAUSS=G2
      ELSE
        SW=1
        A=SQRT(-2.0*ALOG(MTH$RANDOM(SEED)))
        B=2.0*ACOS(-1.0)*(MTH$RANDOM(SEED))
        G2=A*COS(B)
        GAUSS=A*SIN(B)
      ENDIF

      END



      SUBROUTINE EKF8(XHAT,XEEST,XPEST,T,DT,COV,Q,R3,
     +                RANGE,THETA,GAMMA,KFLAG,RES,UPDATE)

C  THIS SUBROUTINE ESTIMATES RELATIVE POSITION AND
C  VELOCITY VECTORS AND ACCELERATION AND MASS FLOW
C  RATE OF THE EVADER USING AN EIGHT STATE EXTENDED
```

```
C  KALMAN FILTER WITH SERIAL UPDATES OF RANGE AND
C  TWO LINE-OF-SIGHT ANGLES.
C  (GELB, 'APPLIED OPTIMAL ESTIMATION',  PP. 182-192)

      REAL*8 XHAT(8),COV(8,8),RANGE,THETA,GAMMA,F(8,8)
      REAL*8 E,E2,E3,E5,U,EDOT,EDOT2,Q(8,8),RES(3)
      REAL*8 EDOT3,T,HYP15,HYP13,H(8)
      REAL*8 XEEST(6),XPEST(6),DT,PDOT(8,8),R,R3(3,3)
      REAL*8 POLD(8,8),DMAT,DCMAT(8),GAIN(8)
      INTEGER I,J,JUP,K,KFLAG,UPDATE

C  INITIALIZE DUMMY VARIABLES
      U=3.986012E14
      E2=XEEST(1)*XEEST(1)+XEEST(3)*XEEST(3)+
     +   XEEST(5)*XEEST(5)
      E=SQRT(E2)
      E3=E2*E
      E5=E3*E2
      EDOT2=XEEST(2)*XEEST(2)+XEEST(4)*XEEST(4)+
     +      XEEST(6)*XEEST(6)
      EDOT=SQRT(EDOT2)
      EDOT3=EDOT*EDOT2

C  COMPUTE F MATRIX
      DO 200 I=1,8
        F(1,I)=0.0
        F(3,I)=0.0
        F(5,I)=0.0
        F(7,I)=0.0
        F(8,I)=0.0
200   CONTINUE

      F(1,2)=1.0
      F(3,4)=1.0
      F(5,6)=1.0

      F(2,1)=(-1.0+3.0*XEEST(1)*XEEST(1)/E2)*U/E3
      F(2,2)=(1.0-XEEST(2)*XEEST(2)/EDOT2)*XHAT(7)/EDOT
      F(2,3)=3.0*U*XEEST(1)*XEEST(3)/E5
      F(2,4)=-XEEST(2)*XHAT(7)*XEEST(4)/EDOT3
      F(2,5)=3.0*U*XEEST(1)*XEEST(5)/E5
      F(2,6)=-XEEST(2)*XHAT(7)*XEEST(6)/EDOT3
      F(2,7)=XEEST(2)/EDOT
      F(2,8)=0.0

      F(4,1)=F(2,3)
      F(4,2)=F(2,4)
      F(4,3)=(-1.0+3.0*XEEST(3)*XEEST(3)/E2)*U/E3
      F(4,4)=(1.0-XEEST(4)*XEEST(4)/EDOT2)*XHAT(7)/EDOT
      F(4,5)=3.0*U*XEEST(3)*XEEST(5)/E5
      F(4,6)=-XEEST(4)*XHAT(7)*XEEST(6)/EDOT3
      F(4,7)=XEEST(4)/EDOT
      F(4,8)=0.0
```

```
      F(6,1)=F(2,5)
      F(6,2)=F(2,6)
      F(6,3)=F(4,5)
      F(6,4)=F(4,6)
      F(6,5)=(-1.0+3.0*XEEST(5)*XEEST(5)/E2)*U/E3
      F(6,6)=(1.0-XEEST(6)*XEEST(6)/EDOT2)*XHAT(7)/EDOT
      F(6,7)=XEEST(6)/EDOT
      F(6,8)=0.0

      F(7,7)=XHAT(8)
      F(7,8)=XHAT(7)

      F(8,8)=XHAT(8)+XHAT(8)

C  PROPAGATE COVARIANCE MATRIX FORWARD (EULER'S METHOD)
C  (USING SYMMETRY, COMPUTE LOWER TRIANGULAR PDOT)
      DO 300 I=1,8
        DO 300 J=1,I
          PDOT(I,J)=Q(I,J)
          DO 300 K=1,8
            PDOT(I,J)=PDOT(I,J)+F(I,K)*COV(K,J)+
     +                COV(I,K)*F(J,K)
300   CONTINUE
      DO 310 I=1,8
        DO 310 J=1,I
          POLD(I,J)=COV(I,J)+(PDOT(I,J)*DT)
310   CONTINUE

C  REASSIGN COV AND ZERO OUT H MATRIX
      DO 320 I=1,8
        COV(I,I)=POLD(I,I)
        H(I)=0.0
        JUP=I-1
        DO 320 J=1,JUP
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
320   CONTINUE

C  PROPAGATE STATE ESTIMATE FORWARD ONE STEP
      CALL RK4SYSE(T,XEEST,DT,XHAT(7),XHAT(8))
      IF (UPDATE .NE. 1) CALL RK4SYSP(T,XPEST,DT)
      DO 330 I=1,6
       XHAT(I)=XEEST(I)-XPEST(I)
330   CONTINUE

      IF (UPDATE .NE. 0) THEN

C  PERFORM RANGE UPDATE
        R=SQRT(XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)+
     +         XHAT(5)*XHAT(5))
        H(1)=XHAT(1)/R
        H(3)=XHAT(3)/R
        H(5)=XHAT(5)/R
```

```
            RES(1)=RANGE-R
            IF (UPDATE .NE. 1) RES(1)=0.0
            CALL UPDATE8(XHAT,H,COV,R3(1,1),RES(1),KFLAG)

C   PERFORM THETA UPDATE
            HYP15=XHAT(1)*XHAT(1)+XHAT(5)*XHAT(5)
            H(1)=-XHAT(5)/HYP15
            H(3)=0.0
            H(5)=XHAT(1)/HYP15
            RES(2)=THETA-ATAN(XHAT(5)/XHAT(1))
            IF (UPDATE .NE. 1) RES(2)=0.0
            CALL UPDATE8(XHAT,H,COV,R3(2,2),RES(2),KFLAG)

C   PERFORM GAMMA UPDATE
            HYP13=XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)
            H(1)=-XHAT(3)/HYP13
            H(3)=XHAT(1)/HYP13
            H(5)=0.0
            RES(3)=GAMMA-ATAN(XHAT(3)/XHAT(1))
            IF (UPDATE .NE. 1) RES(3)=0.0
            CALL UPDATE8(XHAT,H,COV,R3(3,3),RES(3),KFLAG)

         ENDIF

         END



         SUBROUTINE UPDATE8(XHAT,H,COV,VAR,RES,KFLAG)

C   THIS SUBROUTINE DOES ONE SERIAL UPDATE FOR THE
C   EIGHT STATE EXTENDED KALMAN FILTER

         REAL*8 XHAT(8),H(8),COV(8,8),VAR,RES
         REAL*8 DMAT,DCMAT(8),GAIN(8),POLD(8,8)
         INTEGER I,J,JUP,KFLAG

C   INITIALIZE COUNT
         COUNT=0

C   COMPUTE MATRIX (1X1) FOR INVERSION
         DO 110 I=1,8
           DCMAT(I)=0.0
           DO 110 J=1,8
             DCMAT(I)=DCMAT(I)+COV(I,J)*H(J)
110      CONTINUE
         DMAT=VAR
         DO 120 I=1,8
           DMAT=DMAT+DCMAT(I)*H(I)
120      CONTINUE
```

```
C  COMPUTE GAIN MATRIX
      DO 140 I=1,8
        GAIN(I)=0.0
        DO 130 J=1,8
          GAIN(I)=GAIN(I)+COV(I,J)*H(J)
130     CONTINUE
        GAIN(I)=GAIN(I)/DMAT
140   CONTINUE

C  UPDATE COVARIANCE MATRIX AND LIMIT THE GAIN TO
C  PREVENT NEGATIVE DIAGONAL COVARIANCES
      DO 150 I=1,8
        DCMAT(I)=0.0
        DO 150 J=1,8
          DCMAT(I)=DCMAT(I)+H(J)*COV(J,I)
150   CONTINUE

      DO 160 I=1,8
        POLD(I,I)=COV(I,I)-GAIN(I)*DCMAT(I)
        IF (POLD(I,I) .LT. 0.0) THEN
          IF (COUNT .LT. 10) THEN
            DO 155 J=1,8
              GAIN(J)=GAIN(J)/2.0
155         CONTINUE
            COUNT=COUNT+1
            KFLAG=KFLAG+1
            GOTO 150
           ELSE
            KFLAG=KFLAG+990
            RETURN
          ENDIF
        ENDIF
        JUP=I-1
        DO 160 J=1,JUP
          POLD(I,J)=COV(I,J)-GAIN(I)*DCMAT(J)
160   CONTINUE

C  UPDATE ESTIMATE
      DO 170 I=1,8
        XHAT(I)=XHAT(I)+GAIN(I)*RES
170   CONTINUE

C  UPDATE COV MATRIX
      DO 175 I=1,8
        COV(I,I)=POLD(I,I)
        JUP=I-1
        DO 175 J=1,JUP
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
175   CONTINUE

      END
```

```
      SUBROUTINE EKF6(XHAT,XEEST,XPEST,T,DT,COV,Q,R3,
     +            RANGE,THETA,GAMMA,KFLAG,RES,UPDATE)

C  THIS SUBROUTINE ESTIMATES RELATIVE POSITION AND
C  VELOCITY VECTORS AND ACCELERATION OF THE EVADER
C  USING A SIX STATE EXTENDED KALMAN FILTER WITH
C  SERIAL UPDATES OF RANGE AND TWO LOS ANGLES.
C  (GELB, 'APPLIED OPTIMAL ESTIMATION',  PP. 182-192)

      REAL*8 XHAT(8),COV(8,8),RANGE,THETA,GAMMA,F(8,8)
      REAL*8 E,E2,E3,E5,U,EDOT,EDOT2,Q(8,8),RES(3)
      REAL*8 EDOT3,T,HYP15,HYP13,H(8)
      REAL*8 XEEST(6),XPEST(6),DT,PDOT(8,8),R,R3(3,3)
      REAL*8 POLD(8,8),DMAT,DCMAT(8),GAIN(8)
      INTEGER I,J,JUP,K,KFLAG,UPDATE

C  INITIALIZE DUMMY VARIABLES
      U=3.986012E14
      E2=XEEST(1)*XEEST(1)+XEEST(3)*XEEST(3)+
     +   XEEST(5)*XEEST(5)
      E=SQRT(E2)
      E3=E2*E
      E5=E3*E2
      EDOT2=XEEST(2)*XEEST(2)+XEEST(4)*XEEST(4)+
     +      XEEST(6)*XEEST(6)
      EDOT=SQRT(EDOT2)
      EDOT3=EDOT*EDOT2

C  COMPUTE F MATRIX
      DO 200 I=1,8
        F(1,I)=0.0
        F(3,I)=0.0
        F(5,I)=0.0
        F(7,I)=0.0
        F(8,I)=0.0
200   CONTINUE

      F(1,2)=1.0
      F(3,4)=1.0
      F(5,6)=1.0

      F(2,1)=(-1.0+3.0*XEEST(1)*XEEST(1)/E2)*U/E3
      F(2,2)=(1.0-XEEST(2)*XEEST(2)/EDOT2)*XHAT(7)/EDOT
      F(2,3)=3.0*U*XEEST(1)*XEEST(3)/E5
      F(2,4)=-XEEST(2)*XHAT(7)*XEEST(4)/EDOT3
      F(2,5)=3.0*U*XEEST(1)*XEEST(5)/E5
      F(2,6)=-XEEST(2)*XHAT(7)*XEEST(6)/EDOT3
      F(2,7)=0.0
      F(2,8)=0.0

      F(4,1)=F(2,3)
      F(4,2)=F(2,4)
      F(4,3)=(-1.0+3.0*XEEST(3)*XEEST(3)/E2)*U/E3
```

```
      F(4,4)=(1.0-XEEST(4)*XEEST(4)/EDOT2)*XHAT(7)/EDOT
      F(4,5)=3.0*U*XEEST(3)*XEEST(5)/E5
      F(4,6)=-XEEST(4)*XHAT(7)*XEEST(6)/EDOT3
      F(4,7)=0.0
      F(4,8)=0.0

      F(6,1)=F(2,5)
      F(6,2)=F(2,6)
      F(6,3)=F(4,5)
      F(6,4)=F(4,6)
      F(6,5)=(-1.0+3.0*XEEST(5)*XEEST(5)/E2)*U/E3
      F(6,6)=(1.0-XEEST(6)*XEEST(6)/EDOT2)*XHAT(7)/EDOT
      F(6,7)=0.0
      F(6,8)=0.0

C  PROPAGATE COVARIANCE MATRIX FORWARD (EULER'S METHOD)
C  (USING SYMMETRY, COMPUTE LOWER TRIANGULAR PDOT)
      DO 300 I=1,6
        DO 300 J=1,I
          PDOT(I,J)=Q(I,J)
          DO 300 K=1,6
            PDOT(I,J)=PDOT(I,J)+F(I,K)*COV(K,J)+
     +                COV(I,K)*F(J,K)
300   CONTINUE
      DO 310 I=1,6
        DO 310 J=1,I
          POLD(I,J)=COV(I,J)+(PDOT(I,J)*DT)
310   CONTINUE

C  REASSIGN COV AND ZERO OUT H MATRIX
      DO 320 I=1,6
        COV(I,I)=POLD(I,I)
        H(I)=0.0
        JUP=I-1
        DO 320 J=1,JUP
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
320   CONTINUE

C  PROPAGATE STATE ESTIMATE FORWARD ONE STEP
      CALL RK4SYSE(T,XEEST,DT,XHAT(7),XHAT(8))
      IF (UPDATE .NE. 1) CALL RK4SYSP(T,XPEST,DT)
      DO 330 I=1,6
       XHAT(I)=XEEST(I)-XPEST(I)
330   CONTINUE

      IF (UPDATE .NE. 0) THEN

C  PERFORM RANGE UPDATE
        R=SQRT(XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)+
     +         XHAT(5)*XHAT(5))
        H(1)=XHAT(1)/R
        H(3)=XHAT(3)/R
        H(5)=XHAT(5)/R
```

```
      RES(1)=RANGE-R
      IF (UPDATE .NE. 1) RES(1)=0.0
      CALL UPDATE6(XHAT,H,COV,R3(1,1),RES(1),KFLAG)

C  PERFORM THETA UPDATE
      HYP15=XHAT(1)*XHAT(1)+XHAT(5)*XHAT(5)
      H(1)=-XHAT(5)/HYP15
      H(3)=0.0
      H(5)=XHAT(1)/HYP15
      RES(2)=THETA-ATAN(XHAT(5)/XHAT(1))
      IF (UPDATE .NE. 1) RES(2)=0.0
      CALL UPDATE6(XHAT,H,COV,R3(2,2),RES(2),KFLAG)

C  PERFORM GAMMA UPDATE
      HYP13=XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)
      H(1)=-XHAT(3)/HYP13
      H(3)=XHAT(1)/HYP13
      H(5)=0.0
      RES(3)=GAMMA-ATAN(XHAT(3)/XHAT(1))
      IF (UPDATE .NE. 1) RES(3)=0.0
      CALL UPDATE6(XHAT,H,COV,R3(3,3),RES(3),KFLAG)

   ENDIF

   END



   SUBROUTINE EKF60(XHAT,XEEST,XPEST,T,DT,COV,Q,R3,
  +                 RANGE,THETA,GAMMA,KFLAG,RES)

C  THIS SUBROUTINE ESTIMATES RELATIVE POSITION AND
C  VELOCITY VECTORS AND ACCELERATION OF THE EVADER
C  USING A SIX STATE EXTENDED KALMAN FILTER WITH
C  SERIAL UPDATES OF RANGE AND TWO LOS ANGLES.
C  (GELB, 'APPLIED OPTIMAL ESTIMATION',  PP. 182-192)

   REAL*8 XHAT(8),COV(8,8),RANGE,THETA,GAMMA,F(8,8)
   REAL*8 E,E2,E3,E5,U,EDOT,EDOT2,Q(8,8),RES(3)
   REAL*8 EDOT3,T,HYP15,HYP13,H(8)
   REAL*8 XEEST(6),XPEST(6),DT,PDOT(8,8),R,R3(3,3)
   REAL*8 POLD(8,8),DMAT,DCMAT(8),GAIN(8)
   INTEGER I,J,JUP,K,KFLAG

C  INITIALIZE DUMMY VARIABLES
   U=3.986012E14
   E2=XEEST(1)*XEEST(1)+XEEST(3)*XEEST(3)+
  +   XEEST(5)*XEEST(5)
   E=SQRT(E2)
   E3=E2*E
   E5=E3*E2
   EDOT2=XEEST(2)*XEEST(2)+XEEST(4)*XEEST(4)+
  +      XEEST(6)*XEEST(6)
```

```
      EDOT=SQRT(EDOT2)
      EDOT3=EDOT*EDOT2

C  COMPUTE F MATRIX
      DO 200 I=1,8
        DO 200 J=1,8
        F(I,J)=0.0
200   CONTINUE

      F(1,2)=1.0
      F(3,4)=1.0
      F(5,6)=1.0

      F(2,2)=(1.0-XEEST(2)*XEEST(2)/EDOT2)*XHAT(7)/EDOT
      F(2,4)=-XEEST(2)*XHAT(7)*XEEST(4)/EDOT3
      F(2,6)=-XEEST(2)*XHAT(7)*XEEST(6)/EDOT3

      F(4,2)=F(2,4)
      F(4,4)=(1.0-XEEST(4)*XEEST(4)/EDOT2)*XHAT(7)/EDOT
      F(4,6)=-XEEST(4)*XHAT(7)*XEEST(6)/EDOT3

      F(6,2)=F(2,6)
      F(6,4)=F(4,6)
      F(6,6)=(1.0-XEEST(6)*XEEST(6)/EDOT2)*XHAT(7)/EDOT

C  PROPAGATE COVARIANCE MATRIX FORWARD (EULER'S METHOD)
C  (USING SYMMETRY, COMPUTE LOWER TRIANGULAR PDOT)
      DO 300 I=1,6
        DO 300 J=1,I
          PDOT(I,J)=Q(I,J)
          DO 300 K=1,6
            PDOT(I,J)=PDOT(I,J)+F(I,K)*COV(K,J)+
     +                COV(I,K)*F(J,K)
300   CONTINUE
      DO 310 I=1,6
        DO 310 J=1,I
          POLD(I,J)=COV(I,J)+(PDOT(I,J)*DT)
310   CONTINUE

C  REASSIGN COV AND ZERO OUT H MATRIX
      DO 320 I=1,6
        COV(I,I)=POLD(I,I)
        H(I)=0.0
        JUP=I-1
        DO 320 J=1,JUP
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
320   CONTINUE

C  PROPAGATE STATE ESTIMATES FORWARD ONE STEP
      XEEST(1)=XEEST(1)+DT*XEEST(2)
      XEEST(3)=XEEST(3)+DT*XEEST(4)
      XEEST(5)=XEEST(5)+DT*XEEST(6)
```

```
      XEEST(2)=XEEST(2)*(1.0+DT*XHAT(7)/EDOT)
      XEEST(4)=XEEST(4)*(1.0+DT*XHAT(7)/EDOT)
      XEEST(6)=XEEST(6)*(1.0+DT*XHAT(7)/EDOT)
      XHAT(7)=XHAT(7)/(1.0-XHAT(8)*DT)
      XHAT(8)=XHAT(8)/(1.0-XHAT(8)*DT)

      DO 330 I=1,6
       XHAT(I)=XEEST(I)-XPEST(I)
330   CONTINUE


C  PERFORM RANGE UPDATE
      R=SQRT(XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)+
     +       XHAT(5)*XHAT(5))
      H(1)=XHAT(1)/R
      H(3)=XHAT(3)/R
      H(5)=XHAT(5)/R
      RES(1)=RANGE-R
      CALL UPDATE6(XHAT,H,COV,R3(1,1),RES(1),KFLAG)

C  PERFORM THETA UPDATE
      HYP15=XHAT(1)*XHAT(1)+XHAT(5)*XHAT(5)
      H(1)=-XHAT(5)/HYP15
      H(3)=0.0
      H(5)=XHAT(1)/HYP15
      RES(2)=THETA-ATAN(XHAT(5)/XHAT(1))
      CALL UPDATE6(XHAT,H,COV,R3(2,2),RES(2),KFLAG)

C  PERFORM GAMMA UPDATE
      HYP13=XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)
      H(1)=-XHAT(3)/HYP13
      H(3)=XHAT(1)/HYP13
      H(5)=0.0
      RES(3)=GAMMA-ATAN(XHAT(3)/XHAT(1))
      CALL UPDATE6(XHAT,H,COV,R3(3,3),RES(3),KFLAG)


      END



      SUBROUTINE UPDATE6(XHAT,H,COV,VAR,RES,KFLAG)

C  THIS SUBROUTINE DOES ONE SERIAL UPDATE FOR THE
C  SIX STATE EXTENDED KALMAN FILTER

      REAL*8 XHAT(8),H(8),COV(8,8),VAR,RES
      REAL*8 DMAT,DCMAT(8),GAIN(8),POLD(8,8)
      INTEGER I,J,JUP,KFLAG

C  INITIALIZE COUNT
      COUNT=0
```

```
C  COMPUTE MATRIX (1X1) FOR INVERSION
      DO 110 I=1,6
        DCMAT(I)=0.0
        DO 110 J=1,6
          DCMAT(I)=DCMAT(I)+COV(I,J)*H(J)
110   CONTINUE
      DMAT=VAR
      DO 120 I=1,6
        DMAT=DMAT+DCMAT(I)*H(I)
120   CONTINUE

C  COMPUTE GAIN MATRIX
      DO 140 I=1,6
        GAIN(I)=0.0
        DO 130 J=1,6
          GAIN(I)=GAIN(I)+COV(I,J)*H(J)
130     CONTINUE
        GAIN(I)=GAIN(I)/DMAT
140   CONTINUE

C  UPDATE COVARIANCE MATRIX AND LIMIT THE GAIN TO
C  PREVENT NEGATIVE DIAGONAL COVARIANCES
      DO 150 I=1,6
        DCMAT(I)=0.0
        DO 150 J=1,6
          DCMAT(I)=DCMAT(I)+H(J)*COV(J,I)
150   CONTINUE

      DO 160 I=1,6
        POLD(I,I)=COV(I,I)-GAIN(I)*DCMAT(I)
        IF (POLD(I,I) .LT. 0.0) THEN
          IF (COUNT .LT. 10) THEN
            DO 155 J=1,6
              GAIN(J)=GAIN(J)/2.0
155         CONTINUE
            COUNT=COUNT+1
            KFLAG=KFLAG+1
            GOTO 150
           ELSE
            KFLAG=KFLAG+990
            RETURN
          ENDIF
        ENDIF
        JUP=I-1
        DO 160 J=1,JUP
          POLD(I,J)=COV(I,J)-GAIN(I)*DCMAT(J)
160   CONTINUE

C  UPDATE ESTIMATE
      DO 170 I=1,6
        XHAT(I)=XHAT(I)+GAIN(I)*RES
170   CONTINUE
```

```
C  UPDATE COV MATRIX
      DO 175 I=1,6
        COV(I,I)=POLD(I,I)
        JUP=I-1
        DO 175 J=1,JUP
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
175   CONTINUE

      END



      SUBROUTINE EKF3(XHAT,XEEST,XPEST,T,DT,COV,Q,R3,
     +                RANGE,THETA,GAMMA,KFLAG,RES)

C  THIS SUBROUTINE ESTIMATES RELATIVE POSITION AND
C  VELOCITY VECTORS AND ACCELERATION OF THE EVADER
C  USING A THREE STATE EXTENDED KALMAN FILTER WITH
C  SERIAL UPDATES OF RANGE AND TWO LOS ANGLES.
C  (GELB, 'APPLIED OPTIMAL ESTIMATION',  PP. 182-192)

      REAL*8 XHAT(8),COV(8,8),RANGE,THETA,GAMMA,F(8,8)
      REAL*8 E,E2,E3,E5,U,EDOT,EDOT2,Q(8,8),RES(3)
      REAL*8 EDOT3,T,HYP15,HYP13,H(8)
      REAL*8 XEEST(6),XPEST(6),DT,PDOT(8,8),R,R3(3,3)
      REAL*8 POLD(8,8),DMAT,DCMAT(8),GAIN(8)
      INTEGER I,J,JUP,K,KFLAG

C  INITIALIZE DUMMY VARIABLES
      U=3.986012E14
      E2=XEEST(1)*XEEST(1)+XEEST(3)*XEEST(3)+
     +   XEEST(5)*XEEST(5)
      E=SQRT(E2)
      E3=E2*E
      E5=E3*E2
      EDOT2=XEEST(2)*XEEST(2)+XEEST(4)*XEEST(4)+
     +      XEEST(6)*XEEST(6)
      EDOT=SQRT(EDOT2)
      EDOT3=EDOT*EDOT2

C  COMPUTE F MATRIX
      DO 200 I=1,8
        DO 200 J=1,8
        F(I,J)=0.0
200   CONTINUE

      F(1,1)=DT*(-1.0+3.0*XEEST(1)*XEEST(1)/E2)*U/E3
      F(1,3)=DT*3.0*U*XEEST(1)*XEEST(3)/E5
      F(1,5)=DT*3.0*U*XEEST(1)*XEEST(5)/E5
```

```
      F(3,1)=F(1,3)
      F(3,3)=DT*(-1.0+3.0*XEEST(3)*XEEST(3)/E2)*U/E3
      F(3,5)=DT*3.0*U*XEEST(3)*XEEST(5)/E5

      F(5,1)=F(1,5)
      F(5,3)=F(3,5)
      F(5,5)=DT*(-1.0+3.0*XEEST(5)*XEEST(5)/E2)*U/E3

C  PROPAGATE COVARIANCE MATRIX FORWARD (EULER'S METHOD)
C  (USING SYMMETRY, COMPUTE LOWER TRIANGULAR PDOT)
      DO 300 I=1,5,2
        DO 300 J=1,I,2
          PDOT(I,J)=Q(I,J)
          DO 300 K=1,5,2
            PDOT(I,J)=PDOT(I,J)+F(I,K)*COV(K,J)+
     +                COV(I,K)*F(J,K)
300   CONTINUE
      DO 310 I=1,5,2
        DO 310 J=1,I,2
          POLD(I,J)=COV(I,J)+(PDOT(I,J)*DT)
310   CONTINUE

C  REASSIGN COV AND ZERO OUT H MATRIX
      DO 320 I=1,5,2
        COV(I,I)=POLD(I,I)
        H(I)=0.0
        H(I+1)=0.0
        JUP=I-2
        DO 320 J=1,JUP,2
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
320   CONTINUE

C  PROPAGATE STATE ESTIMATES FORWARD ONE STEP
      CALL RK4SYSE(T,XEEST,DT,XHAT(7),XHAT(8))
      DO 330 I=1,6
       XHAT(I)=XEEST(I)-XPEST(I)
330   CONTINUE

C  PERFORM RANGE UPDATE
      R=SQRT(XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)+
     +       XHAT(5)*XHAT(5))
      H(1)=XHAT(1)/R
      H(3)=XHAT(3)/R
      H(5)=XHAT(5)/R
      RES(1)=RANGE-R
      CALL UPDATE3(XHAT,H,COV,R3(1,1),RES(1),KFLAG)

C  PERFORM THETA UPDATE
      HYP15=XHAT(1)*XHAT(1)+XHAT(5)*XHAT(5)
      H(1)=-XHAT(5)/HYP15
      H(3)=0.0
      H(5)=XHAT(1)/HYP15
```

```
      RES(2)=THETA-ATAN(XHAT(5)/XHAT(1))
      CALL UPDATE3(XHAT,H,COV,R3(2,2),RES(2),KFLAG)

C  PERFORM GAMMA UPDATE
      HYP13=XHAT(1)*XHAT(1)+XHAT(3)*XHAT(3)
      H(1)=-XHAT(3)/HYP13
      H(3)=XHAT(1)/HYP13
      H(5)=0.0
      RES(3)=GAMMA-ATAN(XHAT(3)/XHAT(1))
      CALL UPDATE3(XHAT,H,COV,R3(3,3),RES(3),KFLAG)


      END



      SUBROUTINE UPDATE3(XHAT,H,COV,VAR,RES,KFLAG)

C  THIS SUBROUTINE DOES ONE SERIAL UPDATE FOR THE
C  THREE STATE EXTENDED KALMAN FILTER

      REAL*8 XHAT(8),H(8),COV(8,8),VAR,RES
      REAL*8 DMAT,DCMAT(8),GAIN(8),POLD(8,8)
      INTEGER I,J,JUP,KFLAG

C  INITIALIZE COUNT
      COUNT=0

C  COMPUTE MATRIX (1X1) FOR INVERSION
      DO 110 I=1,5,2
        DCMAT(I)=0.0
        DO 110 J=1,5,2
          DCMAT(I)=DCMAT(I)+COV(I,J)*H(J)
110   CONTINUE
      DMAT=VAR
      DO 120 I=1,6
        DMAT=DMAT+DCMAT(I)*H(I)
120   CONTINUE

C  COMPUTE GAIN MATRIX
      DO 140 I=1,5,2
        GAIN(I)=0.0
        GAIN(I+1)=0.0
        DO 130 J=1,5,2
          GAIN(I)=GAIN(I)+COV(I,J)*H(J)
130     CONTINUE
        GAIN(I)=GAIN(I)/DMAT
140   CONTINUE
```

```
C  UPDATE COVARIANCE MATRIX AND LIMIT THE GAIN TO
C  PREVENT NEGATIVE DIAGONAL COVARIANCES
      DO 150 I=1,5,2
        DCMAT(I)=0.0
        DO 150 J=1,5,2
          DCMAT(I)=DCMAT(I)+H(J)*COV(J,I)
150   CONTINUE

      DO 160 I=1,5,2
        POLD(I,I)=COV(I,I)-GAIN(I)*DCMAT(I)
        IF (POLD(I,I) .LT. 0.0) THEN
          IF (COUNT .LT. 10) THEN
            DO 155 J=1,5,2
              GAIN(J)=GAIN(J)/2.0
155         CONTINUE
            COUNT=COUNT+1
            KFLAG=KFLAG+1
            GOTO 150
           ELSE
            KFLAG=KFLAG+990
            RETURN
          ENDIF
        ENDIF
        JUP=I-2
        DO 160 J=1,JUP,2
          POLD(I,J)=COV(I,J)-GAIN(I)*DCMAT(J)
160   CONTINUE

C  UPDATE ESTIMATE
      XHAT(1)=XHAT(1)+GAIN(1)*RES
      XHAT(3)=XHAT(3)+GAIN(3)*RES
      XHAT(5)=XHAT(5)+GAIN(5)*RES

C  UPDATE COV MATRIX
      DO 175 I=1,5,2
        COV(I,I)=POLD(I,I)
        JUP=I-2
        DO 175 J=1,JUP,2
          COV(I,J)=POLD(I,J)
          COV(J,I)=POLD(I,J)
175   CONTINUE

      END
```

```
C     TOOL4

C  THIS IS A COLLECTION OF SUBROUTINES NEEDED FOR
C  SPLINE COMPUTATION AND NUMERICAL SEARCH OF
C  OPTIMAL SOLUTION FOR THE HYPERVELOCITY ORBITAL
C  INTERCEPT PROGRAM


C  SUBROUTINE DICTIONARY
C     A           PRESENT ACCELERATION OF EVADER
C     ACOEF       A SPLINE COEFFICIENT
C     AD          DUMMY ACCELERATION
C     ASIGX       A SPLINE COEFFICIENT OF SIGMAX
C     ASIGY       A SPLINE COEFFICIENT OF SIGMAY
C     ASIGZ       A SPLINE COEFFICIENT OF SIGMAZ
C     AX          A SPLINE COEEFICIENT OF X
C     AY          A SPLINE COEEFICIENT OF Y
C     AZ          A SPLINE COEEFICIENT OF Z
C     BCOEF       B SPLINE COEFFICIENT
C     BSIGX       B SPLINE COEFFICIENT OF SIGMAX
C     BSIGY       B SPLINE COEFFICIENT OF SIGMAY
C     BSIGZ       B SPLINE COEFFICIENT OF SIGMAZ
C     BX          B SPLINE COEEFICIENT OF X
C     BY          B SPLINE COEEFICIENT OF Y
C     BZ          B SPLINE COEEFICIENT OF Z
C     CCOEF       C SPLINE COEFFICIENT
C     CFLAG       CONVERGENCE FLAG
C     COST1       DUAL CONTROL COST
C     COST2       DUAL CONTROL COST
C     CSIGX       C SPLINE COEFFICIENT OF SIGMAX
C     CSIGY       C SPLINE COEFFICIENT OF SIGMAY
C     CSIGZ       C SPLINE COEFFICIENT OF SIGMAZ
C     CVDD        COVARIANCE MATRIX FOR EKF
C     CVDR        COVARIANCE MATRIX FOR EKF
C     CVDY        COVARIANCE MATRIX FOR EKF
C     CVDZ        COVARIANCE MATRIX FOR EKF
C     CVD2        COVARIANCE MATRIX FOR EKF
C     CVDIFY      SELECTED COVARIANCE DIFFERENCE
C     CVDIFZ      SELECTED COVARIANCE DIFFERENCE
C     CVTOT       SELECTED COVARIANCE TOTAL
C     CX          C SPLINE COEEFICIENT OF X
C     CY          C SPLINE COEEFICIENT OF Y
C     CZ          C SPLINE COEEFICIENT OF Z
C     DA          DISTANCE DUE TO BOOSTER ACCELERATION
C     DCOEF       D SPLINE COEFFICIENT
C     DDY         CHANGE IN Y VELOCITY
C     DDZ         CHANGE IN Z VELOCITY
C     DELTAY      CHANGE IN Y VELOCITY
C     DELTAZ      CHANGE IN Z VELOCITY
C     DEN         DENOMINATOR (FOR JACOBIAN DETERMINANT)
C     DSIGX       D SPLINE COEFFICIENT OF SIGMAX
C     DSIGY       D SPLINE COEFFICIENT OF SIGMAY
C     DSIGZ       D SPLINE COEFFICIENT OF SIGMAZ
```

```
C       DTGO        CHANGE IN TIME-TO-GO
C       DUMMY       DUMMY VARIABLE
C       DVY         Y VELOCITY CHANGE FOR DEVIATION
C       DVYA        Y VELOCITY CHANGE FOR DEVIATION
C       DVZ         Z VELOCITY CHANGE FOR DEVIATION
C       DVZA        Z VELOCITY CHANGE FOR DEVIATION
C       DX          D SPLINE COEEFICIENT OF X
C       DY          D SPLINE COEEFICIENT OF Y
C       DZ          D SPLINE COEEFICIENT OF Z
C       FSIGX       SIGMA X SPLINE COEFF FOR Y VEL CHANGE
C       FSIGY       SIGMA Y SPLINE COEFF FOR Y VEL CHANGE
C       FSIGZ       SIGMA Z SPLINE COEFF FOR Y VEL CHANGE
C       F1          NONLINEAR SYSTEM EQUATION VALUE
C       F2          NONLINEAR SYSTEM EQUATION VALUE
C       F3          NONLINEAR SYSTEM EQUATION VALUE
C       GSIGX       SIGMA X SPLINE COEFF FOR Z VEL CHANGE
C       GSIGY       SIGMA Y SPLINE COEFF FOR Z VEL CHANGE
C       GSIGZ       SIGMA Z SPLINE COEFF FOR Z VEL CHANGE
C       H           ITERATION TIME STEP SIZE
C       I           COUNTER
C       J           COUNTER
C       J11         JACOBIAN MATRIX 1,1 ELEMENT
C       J12         JACOBIAN MATRIX 1,2 ELEMENT
C       J13         JACOBIAN MATRIX 1,3 ELEMENT
C       J22         JACOBIAN MATRIX 2,2 ELEMENT
C       LAMBDA      LAGRANGE MULTIPLIER
C       MAXDV       MAXIMUM PERMISSIBLE VELOCITY CHANGE
C       MISS        MISS DISTANCE
C       MDOT        UNITIZED MASS FLOW RATE OF EVADER
C       OLDTGO      PREVIOUS TIME-TO-GO
C       RADICL      QUANTITY INSIDE RADICAL SIGN
C       RANGE       RELATIVE RANGE
C       SIG         DEVIATION AT FINAL TIME
C       SIGMAM      MEASUREMENT DEVIATIONS
C       SIGX        X DEVIATION AT FINAL TIME
C       SIGY        Y DEVIATION AT FINAL TIME
C       SIGZ        Z DEVIATION AT FINAL TIME
C       SIGXDT      TIME DERIVATIVE OF SIGX
C       SIGYDT      TIME DERIVATIVE OF SIGY
C       SIGZDT      TIME DERIVATIVE OF SIGZ
C       TD          DUMMY TIME
C       TGO         TIME-TO-GO
C       TGO3        3.0*TGO
C       TGO6        6.0*TGO
C       TLIM        TIME STEP LIMIT
C       TOL         MISS TOLERANCE
C       VA          VELOCITY DUE TO BOOSTER ACCELERATION
C       VE          EVADER VELOCITY MAGNITUDE
C       VX          EVADER UNITIZED X VELOCITY
C       VXE         EVADER X VELOCITY
C       VY          EVADER UNITIZED Y VELOCITY
C       VYE         EVADER Y VELOCITY
C       VZ          EVADER UNITIZED Z VELOCITY
```

```
C      VZE          EVADER Z VELOCITY
C      XDUALD       EKF STATE ESTIMATE
C      XDUALR       EKF STATE ESTIMATE
C      XDUALY       EKF STATE ESTIMATE
C      XDUALZ       EKF STATE ESTIMATE
C      XDUAL2       EKF STATE ESTIMATE
C      XE           EVADER RELATIVE STATE VECTOR
C      XEDD         EVADER RELATIVE STATE VECTOR
C      XEDR         EVADER RELATIVE STATE VECTOR
C      XEDY         EVADER RELATIVE STATE VECTOR
C      XEDZ         EVADER RELATIVE STATE VECTOR
C      XED2         EVADER RELATIVE STATE VECTOR
C      XF           FINAL RELATIVE X POSITION
C      XFDOT        FINAL RELATIVE X VELOCITY
C      XP           PURSUER RELATIVE STATE VECTOR
C      XPDD         PURSUER RELATIVE STATE VECTOR
C      XPDR         PURSUER RELATIVE STATE VECTOR
C      XPDY         PURSUER RELATIVE STATE VECTOR
C      XPDZ         PURSUER RELATIVE STATE VECTOR
C      XPD2         PURSUER RELATIVE STATE VECTOR
C      XR           RELATIVE STATE VECTOR
C      X0           INITIAL RELATIVE X POSITION
C      X1           RELATIVE X POSITION AT ONE THIRD POINT
C      X2           RELATIVE X POSITION AT TWO THIRDS POINT
C      X3           RELATIVE X POSITION AT FINAL POINT
C      X0DOT        INITIAL RELATIVE X VELOCITY
C      YF           FINAL RELATIVE Y POSITION
C      YFDOT        FINAL RELATIVE Y VELOCITY
C      ZF           FINAL RELATIVE Z POSITION
C      ZFDOT        FINAL RELATIVE Z VELOCITY



      SUBROUTINE SPLINE4(X0,X1,X2,X3,TGO,
     +                   ACOEF,BCOEF,CCOEF,DCOEF)

C  THIS SUBROUTINE CREATES THE SPLINE COEFFICIENTS
C  FOR THE EQUATION:
C       ACOEF*T**3 + BCOEF*T**2 + CCOEF*T + DCOEF
C  GIVEN FOUR EQUISPACED POINTS AND TIME-TO-GO.
C  PRESENT TIME IS EQUAL TO ZERO.

      REAL*8 X0,X1,X2,X3,TGO,ACOEF,BCOEF,CCOEF,DCOEF

      ACOEF=(-4.5*X0 +13.5*X1 -13.5*X2 +4.5*X3)/
     +       (TGO*TGO*TGO)
      BCOEF=(+9.0*X0 -22.5*X1 +18.0*X2 -4.5*X3)/
     +       (TGO*TGO)
      CCOEF=(-5.5*X0  +9.0*X1  -4.5*X2 +1.0*X3)/TGO
      DCOEF=X0

      END
```

```
      SUBROUTINE SPLINE(X0,X0DOT,XF,XFDOT,TGO,
     +                  ACOEF,BCOEF,CCOEF,DCOEF)

C  THIS SUBROUTINE CREATES THE SPLINE COEFFICIENTS
C  FOR THE EQUATION:
C       ACOEF*T**3 + BCOEF*T**2 + CCOEF*T + DCOEF
C  GIVEN INITIAL POSITION AND VELOCITY AND
C  FINAL POSITION, VELOCITY, AND TIME-TO-GO.
C  PRESENT TIME IS EQUAL TO ZERO.

      REAL*8 X0,X0DOT,XF,XFDOT,TGO
      REAL*8 ACOEF,BCOEF,CCOEF,DCOEF

      DCOEF=X0
      CCOEF=X0DOT
      ACOEF=(2.0*(X0-XF)/TGO+X0DOT+XFDOT)/TGO/TGO
      BCOEF=(3.0*(XF-X0)/TGO-2.0*X0DOT-XFDOT)/TGO

      END



      SUBROUTINE SEARCHA(AX,BX,CX,DX,AY,BY,CY,DY,
     +  AZ,BZ,CZ,DZ,K,TGO,DELTAY,DELTAZ,TOL,CFLAG)

C  THIS SUBROUTINE NUMERICALLY MINIMIZES THE
C   COST FUNCTION:
C        L = K*(XF**2+XY**2+XZ**2)/2 +
C            (DELTAX**2 + DELTAY**2)/2
C  BY VARYING THE TIME-TO-GO (TGO) AND THE VELOCITY
C  CHANGES (DELTAY AND DELTAZ) TO BRING THE MISS
C  DISTANCE WITHIN SOME TOLERANCE.
C  XF,YF AND ZF ARE COMPUTED FROM THE SPLINE
C  COEFFICIENTS AX,BX,...,DZ AT THE FINAL TIME.
C  THIS IS ACCOMPLISHED BY EMPLOYING A NEWTON-RAPHSON
C  SEARCH SCHEME FOR NON-LINEAR SYSTEMS.
C   (PP. 176-179 OF MARON, 'NUMERICAL ANALYSIS,
C    A PRACTICAL APPROACH')

      REAL*8 AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,CZ,DZ,DDY
      REAL*8 K,TGO,DELTAY,DELTAZ,XF,YF,ZF,F1,F2,F3
      REAL*8 XFDOT,YFDOT,ZFDOT,J11,J12,J13,J22,TGO3
      REAL*8 MISS,TOL,TGO6,OLDTGO,SCALE,DEN,DTGO,DDZ
      INTEGER I,CFLAG

C  INITIALIZE VELOCITY CHANGES AND OLD FINAL TIME
      DELTAY=0.0
      DELTAZ=0.0
      OLDTGO=TGO

C  BEGIN SEARCH LOOP
      DO 10 I=1,25
```

```
C  COMPUTE FINAL RELATIVE POSITIONS
      XF=((AX*TGO+BX)*TGO+CX)*TGO+DX
      YF=((AY*TGO+BY)*TGO+CY-DELTAY)*TGO+DY
      ZF=((AZ*TGO+BZ)*TGO+CZ-DELTAZ)*TGO+DZ
      MISS=SQRT(XF*XF+YF*YF+ZF*ZF)

C  TEST FOR NEARNESS (THIS IS RANGE RELATIVE)
      IF (MISS .LE. TOL) RETURN

C  COMPUTE FINAL RELATIVE VELOCITIES
      TGO3=TGO+TGO+TGO
      XFDOT=(AX*TGO3+BX+BX)*TGO+CX
      YFDOT=(AY*TGO3+BY+BY)*TGO+CY-DELTAY
      ZFDOT=(AZ*TGO3+BZ+BZ)*TGO+CZ-DELTAZ

C  COMPUTE NONLINEAR SYSTEM EQUATIONS
      F1=K*(XF*XFDOT+YF*YFDOT+ZF*ZFDOT)
      F2=DELTAY-K*YF*TGO
      F3=DELTAZ-K*ZF*TGO

C  COMPUTE NECESSARY ELEMENTS OF JACOBIAN MATRIX
      TGO6=TGO3+TGO3
      J11=K*(XF*(AX*TGO6+BX+BX)+YF*(AY*TGO6+BY+BY)+
     +     ZF*(AZ*TGO6+BZ+BZ)+XFDOT*XFDOT+
     +     YFDOT*YFDOT+ZFDOT*ZFDOT)
      J12=-K*(YF+YFDOT*TGO)
      J13=-K*(ZF+ZFDOT*TGO)
      J22=1.0+K*TGO*TGO

C  COMPUTE CHANGES IN CONTROL PARAMETERS
      DEN=(J11*J22-J13*J13-J12*J12)
      DTGO=-(F1*J22-J12*F2-J13*F3)/DEN
      DDY=-(-J12*F1+(J11-J13*J13/J22)*F2+
     +       F3*J12*J13/J22)/DEN
      DDZ=-(-J13*F1+F2*J12*J13/J22+
     +       (J11-J12*J12/J22)*F3)/DEN

C  UPDATE CONTROL PARAMETERS
      TGO=TGO+DTGO
      DELTAY=DELTAY+DDY
      DELTAZ=DELTAZ+DDZ

C  TEST FOR CONVERGENCE OF FINAL TIME
      IF (ABS(DTGO) .LT. (.000001*TGO)) RETURN
10    CONTINUE

C  INCREMENT CONVERGENCE FLAG
      CFLAG=CFLAG+1

C  ZERO OUT VELOCITY CHANGES AND RESET TIME
      DELTAY=0.0
      DELTAZ=0.0
      TGO=OLDTGO
      END
```

```
      SUBROUTINE SEARCHB(AX,BX,CX,DX,AY,BY,CY,DY,
     +    AZ,BZ,CZ,DZ,TGO,DELTAY,DELTAZ,TOL,CFLAG)

C  THIS SUBROUTINE FINDS THE TIME-TO-GO AND VELOCITY
C  CHANGES WITH MISS DISTANCE WITHIN SOME EXTERNAL
C  TOLERANCE.
C  THIS IS DONE BY VARYING THE TIME-TO-GO (TGO) TO
C  VARY THE X MISS DISTANCE.
C  THIS IS ACCOMPLISHED BY EMPLOYING A NEWTON-RAPHSON
C  SEARCH SCHEME (PP. 48-53 OF MARON, 'NUMERICAL
C  ANALYSIS, A PRACTICAL APPROACH')
C  XF,AND XFDOT ARE COMPUTED FROM THE SPLINE
C  COEFFICIENTS AX,BX,CX,DX AT THE FINAL TIME.
C  ONCE THE FINAL TIME IS KNOWN THE VELOCITY CHANGES
C  (DELTAY AND DELTAZ) ARE COMPUTED.

      REAL*8 AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,CZ,DZ,DTGO
      REAL*8 TGO,DELTAY,DELTAZ,XFDOT,SCALE,XF,YF,ZF
      REAL*8 MISS,TOL,DDY,DDZ,OLDTGO
      INTEGER I,CFLAG

C  INITIALIZE VARIABLES
      DELTAY=0.0
      DELTAZ=0.0
      OLDTGO=TGO

C  BEGIN SEARCH LOOP
      DO 10 I=1,10

C  COMPUTE FINAL POSITIONS AND MISS DISTANCE
        XF=((AX*TGO+BX)*TGO+CX)*TGO+DX
        YF=((AY*TGO+BY)*TGO+CY-DELTAY)*TGO+DY
        ZF=((AZ*TGO+BZ)*TGO+CZ-DELTAZ)*TGO+DZ
        MISS=SQRT(XF*XF+YF*YF+ZF*ZF)

C  TEST FOR NEARNESS
        IF (MISS .LE. TOL) RETURN

C  COMPUTE CHANGE IN TIME-TO-GO
        XFDOT=(AX*(TGO+TGO+TGO)+BX+BX)*TGO+CX
        DTGO=-XF/XFDOT
        TGO=TGO+DTGO

C  COMPUTE CHANGES IN VELOCITIES
        YF=((AY*TGO+BY)*TGO+CY-DELTAY)*TGO+DY
        ZF=((AZ*TGO+BZ)*TGO+CZ-DELTAZ)*TGO+DZ
        DDY=(AY*TGO+BY)*TGO+CY-DELTAY+DY/TGO
        DELTAY=DELTAY+DDY
        DDZ=(AZ*TGO+BZ)*TGO+CZ-DELTAZ+DZ/TGO
        DELTAZ=DELTAZ+DDZ

C  TEST FOR CONVERGENCE OF FINAL TIME
        IF (ABS(DTGO) .LT. (.000001*TGO)) RETURN
```

```
10    CONTINUE

C  INCREMENT CONVERGENCE FLAG
      CFLAG=CFLAG+1

C  ZERO OUT VELOCITY CHANGES AND RESET TIME
      DELTAY=0.0
      DELTAZ=0.0
      TGO=OLDTGO

      END



      SUBROUTINE SEARCHC(XR,VXE,VYE,VZE,A,MDOT,
     +                   TGO,DELTAY,DELTAZ,TOL,CFLAG)

C  THIS SUBROUTINE DETERMINES TIME-TO-GO AND VELOCITY
C  CHANGES FOR PLAN C WITH MISS DISTANCE WITHIN SOME
C  TOLERANCE.
C  THIS IS ACCOMPLISHED BY EMPLOYING A NEWTON-RAPHSON
C  SEARCH SCHEME (PP. 48-53 OF MARON, 'NUMERICAL
C  ANALYSIS, A PRACTICAL APPROACH')
C  ONCE THE TIME-TO-GO IS KNOWN THE VELOCITY CHANGES
C  (DELTAY AND DELTAZ) ARE COMPUTED.

      REAL*8 DTGO,XR(6),VXE,VYE,VZE,A,MDOT,DA,VE
      REAL*8 TF,DELTAY,DELTAZ,XF,YF,ZF,XFDOT,SCALE
      REAL*8 OLDTGO,VX,VY,VZ,MISS,TGO,TOL,MT
      INTEGER I,J,CFLAG

C  COMPUTE EVADER UNITIZED VELOCITY VECTOR COMPONENTS
      VE=SQRT(VXE*VXE+VYE*VYE+VZE*VZE)
      VX=VXE/VE
      VY=VYE/VE
      VZ=VZE/VE

C  INITIALIZE VARIABLES
      DELTAY=0.0
      DELTAZ=0.0
      OLDTGO=TGO

C  COMPUTE DISTANCE DUE TO BOOSTER ACCELERATION
      DA=0.0
      MT=MDOT*TGO
      DO 15 J=2,50
        DA=DA+MT**J/(J*(J-1))
15    CONTINUE
      DA=DA*A/(MDOT*MDOT)

C  BEGIN SEARCH LOOP
      DO 30 I=1,10
```

```
C  COMPUTE FINAL RELATIVE POSITION AND MISS DISTANCE
      XF=XR(1)+XR(2)*TGO+DA*VX
      YF=XR(3)+(XR(4)-DELTAY)*TGO+DA*VY
      ZF=XR(5)+(XR(6)-DELTAZ)*TGO+DA*VZ
      MISS=SQRT(XF*XF+YF*YF+ZF*ZF)

C  TEST FOR NEARNESS
      IF (MISS .LE. TOL) RETURN

C  COMPUTE CHANGE IN TIME-TO-GO
      XFDOT=XR(2)+DA*VX/TGO
      DTGO=-XF/XFDOT
      TGO=TGO+DTGO

C  COMPUTE CHANGES IN VELOCITIES
      DA=0.0
      MT=MDOT*TGO
      DO 25 J=2,50
        DA=DA+MT**J/(J*(J-1))
25    CONTINUE
      DA=DA*A/(MDOT*MDOT)
      DELTAY=(DA*VY+XR(3))/TGO+XR(4)
      DELTAZ=(DA*VZ+XR(5))/TGO+XR(6)

C  TEST FOR CONVERGENCE OF TIME-TO-GO
      IF (ABS(DTGO) .LT. (.000001*TGO)) RETURN

30    CONTINUE

C  INCREMENT CONVERGENCE FLAG
      CFLAG=CFLAG+1

C  ZERO OUT VELOCITY CHANGES AND RESET TIME
      DELTAY=0.0
      DELTAZ=0.0
      TGO=OLDTGO

      END



      SUBROUTINE SEARCHCC(AX,BX,CX,DX,AY,BY,CY,DY,
     +   AZ,BZ,CZ,DZ,ASIGX,BSIGX,CSIGX,DSIGX,ASIGY,
     +   BSIGY,CSIGY,DSIGY,ASIGZ,BSIGZ,CSIGZ,DSIGZ,
     +   K,TGO,DELTAY,DELTAZ,TOL,CFLAG)

C  THIS SUBROUTINE NUMERICALLY MINIMIZES THE
C  COST FUNCTION:
C        L = (DELTAX**2 + DELTAY**2)/2
C    SUBJECT TO THE CONSTRAINT :
C        XF*XF+YF*YF+ZF*ZF <= K*(SIGMAF*SIGMAF)
```

```
C  BY VARYING THE TIME-TO-GO (TGO) AND THE VELOCITY
C  CHANGES (DELTAY AND DELTAZ).
C  XF,YF AND ZF ARE COMPUTED FROM THE SPLINE
C  COEFFICIENTS AX,BX,...,DZ AT THE FINAL TIME,
C  AS ARE THE FINAL SIGMAS.
C  THIS IS ACCOMPLISHED BY EMPLOYING A NEWTON-RAPHSON
C  SEARCH SCHEME FOR NON-LINEAR SYSTEMS.
C   (PP. 176-179 OF MARON, 'NUMERICAL ANALYSIS,
C    A PRACTICAL APPROACH')

      REAL*8 AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,CZ,DZ
      REAL*8 DTGO,TLIM,SCALE,DEN,OLDTGO,KSIG
      REAL*8 K,TGO,DELTAY,DELTAZ,XF,YF,ZF,F1,F2,F3
      REAL*8 XFDOT,YFDOT,ZFDOT,J11,J12,J21,J22,TGO3
      REAL*8 ASIGX,BSIGX,CSIGX,DSIGX,ASIGY,BSIGY
      REAL*8 CSIGY,DSIGY,DLAM,PERT,MISS,F1T,F2T,F1L,F2L
      REAL*8 TOL,ASIGZ,BSIGZ,CSIGZ,DSIGZ,LAMBDA
      REAL*8 SIGX,SIGY,SIGZ,SIGXDT,SIGYDT,SIGZDT
      REAL*8 DYDL,DZDL,DYDTDL,DZDTDL,TGO6,YS,ZS
      REAL*8 SIGXDD,SIGYDD,SIGZDD,XFDD,YFDD,ZFDD
      INTEGER I,J,CFLAG

C  INITIALIZE PARAMETERS
      PERT=0.0000001
      OLDTGO=TGO
      TLIM=0.01*TGO
      DELTAY=0.0
      DELTAZ=0.0

C  DETERMINE IF CONSTRAINT CAN BE SATISFIED
C  WITHOUT THRUSTING
      DO 5 I=1,10

C  COMPUTE FINAL RELATIVE POSITIONS AND DEVIATIONS
        XF=((AX*TGO+BX)*TGO+CX)*TGO+DX
        YF=((AY*TGO+BY)*TGO+CY)*TGO+DY
        ZF=((AZ*TGO+BZ)*TGO+CZ)*TGO+DZ
        SIGX=((ASIGX*TGO+BSIGX)*TGO+CSIGX)*TGO+DSIGX
        SIGY=((ASIGY*TGO+BSIGY)*TGO+CSIGY)*TGO+DSIGY
        SIGZ=((ASIGZ*TGO+BSIGZ)*TGO+CSIGZ)*TGO+DSIGZ

C  COMPUTE FINAL RELATIVE VELOCITIES
C  AND DEVIATION DERIVATIVES
        TGO3=TGO+TGO+TGO
        XFDOT=(AX*TGO3+BX+BX)*TGO+CX
        YFDOT=(AY*TGO3+BY+BY)*TGO+CY
        ZFDOT=(AZ*TGO3+BZ+BZ)*TGO+CZ
        SIGXDT=(ASIGX*TGO3+BSIGX+BSIGX)*TGO+CSIGX
        SIGYDT=(ASIGY*TGO3+BSIGY+BSIGY)*TGO+CSIGY
        SIGZDT=(ASIGZ*TGO3+BSIGZ+BSIGZ)*TGO+CSIGZ
```

```
C  COMPUTE NONLINEAR SYSTEM EQUATIONS
        F1=(XF*XF+YF*YF+ZF*ZF-K*(SIGX*SIGX+
     +     SIGY*SIGY+SIGZ*SIGZ))/2.0
C       DETERMINE IF CONSTRAINT IS SATISFIED
        IF (F1 .LT. TOL) RETURN
        F2=XF*XFDOT+YF*YFDOT+ZF*ZFDOT-K*(SIGX*SIGXDT+
     +     SIGY*SIGYDT+SIGZ*SIGZDT)
        F3=XFDOT*XFDOT+YFDOT*YFDOT+ZFDOT*ZFDOT-K*
     +     (SIGXDT*SIGXDT+SIGYDT*SIGYDT+SIGZDT*SIGZDT)

C  COMPUTE NEW TGO
        DTGO=-F2/F3
        TGO=TGO+DTGO

C  TEST FOR CONVERGENCE OF FINAL TIME
        IF (ABS(DTGO) .LT. (TOL*TLIM)) GOTO 6

5       CONTINUE
6       CONTINUE

C  INITIALIZE LAMBDA
        MISS=SQRT(XF*XF+YF*YF+ZF*ZF)
        KSIG=SQRT(K*(SIGX*SIGX+SIGY*SIGY+SIGZ*SIGZ))
        LAMBDA=(1.0-KSIG/MISS)/TGO/TGO

C  BEGIN SEARCH LOOP
        DO 15 I=1,20

C  COMPUTE VELOCITY CHANGES
        YS=((AY*TGO+BY)*TGO+CY)*TGO+DY
        ZS=((AZ*TGO+BZ)*TGO+CZ)*TGO+DZ
        DEN=1.0+LAMBDA*TGO*TGO
        DELTAY=YS*TGO*LAMBDA/DEN
        DELTAZ=ZS*TGO*LAMBDA/DEN

C  COMPUTE FINAL RELATIVE POSITIONS AND DEVIATIONS
        XF=((AX*TGO+BX)*TGO+CX)*TGO+DX
        YF=YS-DELTAY*TGO
        ZF=ZS-DELTAZ*TGO
        SIGX=((ASIGX*TGO+BSIGX)*TGO+CSIGX)*TGO+DSIGX
        SIGY=((ASIGY*TGO+BSIGY)*TGO+CSIGY)*TGO+DSIGY
        SIGZ=((ASIGZ*TGO+BSIGZ)*TGO+CSIGZ)*TGO+DSIGZ

C  COMPUTE FINAL RELATIVE VELOCITIES
C  AND DEVIATION DERIVATIVES
        TGO3=TGO+TGO+TGO
        XFDOT=(AX*TGO3+BX+BX)*TGO+CX
        YFDOT=(AY*TGO3+BY+BY)*TGO+CY-DELTAY
        ZFDOT=(AZ*TGO3+BZ+BZ)*TGO+CZ-DELTAZ
        SIGXDT=(ASIGX*TGO3+BSIGX+BSIGX)*TGO+CSIGX
        SIGYDT=(ASIGY*TGO3+BSIGY+BSIGY)*TGO+CSIGY
        SIGZDT=(ASIGZ*TGO3+BSIGZ+BSIGZ)*TGO+CSIGZ
```

```
C  COMPUTE NONLINEAR SYSTEM EQUATIONS
      F1=(XF*XF+YF*YF+ZF*ZF-K*(SIGX*SIGX+
     +       SIGY*SIGY+SIGZ*SIGZ))/2.0
      F2=XF*XFDOT+YF*YFDOT+ZF*ZFDOT-K*
     +       (SIGX*SIGXDT+SIGY*SIGYDT+SIGZ*SIGZDT)

C  TEST FOR CONVERGENCE OF CONSTRAINTS
      IF ((ABS(F1) .LT. TOL) .AND.
     +       (ABS(F2) .LT. TOL)) RETURN

C  COMPUTE NECESSARY PARTIALS
      DYDTDL=-YF*TGO/DEN
      DZDTDL=-ZF*TGO/DEN
      DYDL=DYDTDL*TGO
      DZDL=DZDTDL*TGO
      TGO6=TGO3+TGO3
      XFDD=AX*TGO6+BX+BX
      YFDD=AY*TGO6+BY+BY-LAMBDA*(YFDOT*TGO+YF)
      ZFDD=AZ*TGO6+BZ+BZ-LAMBDA*(ZFDOT*TGO+ZF)
      SIGXDD=ASIGX*TGO6+BSIGX+BSIGX
      SIGYDD=ASIGY*TGO6+BSIGY+BSIGY
      SIGZDD=ASIGZ*TGO6+BSIGZ+BSIGZ

C  COMPUTE NECESSARY ELEMENTS OF JACOBIAN MATRIX
      J11=F2
      J12=YF*DYDL+ZF*DZDL
      J21=XF*XFDD+XFDOT*XFDOT+YF*YFDD+YFDOT*YFDOT+
     +       ZF*ZFDD+ZFDOT*ZFDOT
      J21=J21-K*(SIGX*SIGXDD+SIGXDT*SIGXDT+
     +       SIGY*SIGYDD+SIGYDT*SIGYDT+SIGZ*SIGZDD+
     +       SIGZDT*SIGZDT)
      J22=DYDL*YFDOT+YF*DYDTDL+DZDL*ZFDOT+ZF*DZDTDL

C  COMPUTE CHANGES IN CONTROL PARAMETERS
      DEN=(J11*J22-J12*J21)
      DTGO=(-F1*J22+F2*J12)/DEN
      DLAM=(F1*J21-F2*J11)/DEN
      SCALE=ABS(DTGO/TLIM)
      IF (SCALE .GT. 1.0) THEN
        DTGO=DTGO/SCALE
        DLAM=DLAM/SCALE
      ENDIF

C  UPDATE CONTROL PARAMETERS
      TGO=TGO+DTGO
      LAMBDA=LAMBDA+DLAM

15    CONTINUE


C  INCREMENT CONVERGENCE FLAG
      CFLAG=CFLAG+1
```

```
C  ZERO OUT VELOCITY CHANGES AND RESET TIME
      DELTAY=0.0
      DELTAZ=0.0
      TGO=OLDTGO

      END



      SUBROUTINE SEARCHD(AX,BX,CX,DX,AY,BY,CY,DY,AZ,
     +  BZ,CZ,DZ,K,TGO,DELTAY,DELTAZ,TOL,CFLAG,XDUALD,
     +  XEDD,XPDD,CVDD,MAXDV,COUNT,Q,R3,SIGMAM,H)

C  THIS SUBROUTINE NUMERICALLY MINIMIZES THE
C  COST FUNCTION:
C        L = E{K*(XF**2+XY**2+XZ**2)/2 +
C              (DELTAX**2 + DELTAY**2)/2}
C  BY VARYING THE TIME-TO-GO (TGO) AND THE VELOCITY
C  CHANGES (DELTAY AND DELTAZ) TO BRING THE MISS
C  DISTANCE WITHIN SOME TOLERANCE.
C  XF,YF AND ZF ARE COMPUTED FROM THE SPLINE
C  COEFFICIENTS AX,BX,...,DZ AT THE FINAL TIME.

      REAL*8 AX,BX,CX,DX,AY,BY,CY,DY,AZ,BZ,CZ,DZ
      REAL*8 K,TGO,DELTAY,DELTAZ,DH,Q(8,8),RES(3)
      REAL*8 DVY,DVZ,XDUALD(8),CVDD(8,8),XPDD(6)
      REAL*8 MISS,TOL,XEDD(6),R3(3,3),TD,RANGE,DVYA
      REAL*8 SIGMAM(3),COST1,COST2,CVTOT,DVZA
      REAL*8 XDUAL2(8),CVD2(8,8),XED2(6),XPD2(6)
      REAL*8 CVDY(8,8),CVDZ(8,8),XEDY(6),XEDZ(6)
      REAL*8 XPDY(6),XPDZ(6),XDUALY(8),XDUALZ(8)
      REAL*8 XF,YF,ZF,XFDOT,YFDOT,ZFDOT,MAXDV,DTGO
      REAL*8 CVDR(8,8),XEDR(6),XPDR(6),XDUALR(8)
      REAL*8 DEN,CVDIFY,CVDIFZ,RADICL,RTGO
      INTEGER I,J,CFLAG,COUNT,KFLAG

C  COMPUTE CERTAINTY EQUIVALENCE SOLUTION
      CALL SEARCHA(AX,BX,CX,DX,AY,BY,CY,DY,
     +  AZ,BZ,CZ,DZ,K,TGO,DELTAY,DELTAZ,TOL,CFLAG)

C  ASSIGN DUMMY FILTER VARIABLES
      DO 100 I=1,8
        XDUAL2(I)=XDUALD(I)
        XDUALY(I)=XDUALD(I)
        XDUALZ(I)=XDUALD(I)
        XDUALR(I)=XDUALD(I)
        DO 100 J=1,8
          CVD2(I,J)=CVDD(I,J)
          CVDY(I,J)=CVDD(I,J)
          CVDZ(I,J)=CVDD(I,J)
          CVDR(I,J)=CVDD(I,J)
100   CONTINUE
```

```
      DO 110 I=1,6
        XED2(I)=XEDD(I)
        XEDY(I)=XEDD(I)
        XEDZ(I)=XEDD(I)
        XEDR(I)=XEDD(I)
        XPD2(I)=XPDD(I)
        XPDY(I)=XPDD(I)
        XPDZ(I)=XPDD(I)
        XPDR(I)=XPDD(I)
110   CONTINUE

C  DETERMINE BEST DIRECTION TO IMPROVE THE ESTIMATE
      DVY=MAXDV
      DVZ=MAXDV
      XPDY(4)=XPDY(4)+DVY
      XPDZ(6)=XPDZ(6)+DVZ
      XDUALY(4)=XDUALY(4)-DVY
      XDUALZ(6)=XDUALZ(6)-DVZ

      DH=TGO/COUNT
      DO 300 I=1,COUNT
        RANGE=SQRT(XDUALR(1)*XDUALR(1)+
     +     XDUALR(3)*XDUALR(3)+XDUALR(5)*XDUALR(5))
        R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
        CALL EKF8(XDUALR,XEDR,XPDR,TD,DH,CVDR,Q,R3,
     +            0.0,0.0,0.0,KFLAG,RES,2)
        RANGE=SQRT(XDUALY(1)*XDUALY(1)+
     +     XDUALY(3)*XDUALY(3)+XDUALY(5)*XDUALY(5))
        R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
        CALL EKF8(XDUALY,XEDY,XPDY,TD,DH,CVDY,Q,R3,
     +            0.0,0.0,0.0,KFLAG,RES,2)
        RANGE=SQRT(XDUALZ(1)*XDUALZ(1)+
     +     XDUALZ(3)*XDUALZ(3)+XDUALZ(5)*XDUALZ(5))
        R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
        CALL EKF8(XDUALZ,XEDZ,XPDZ,TD,DH,CVDZ,Q,R3,
     +            0.0,0.0,0.0,KFLAG,RES,2)
300   CONTINUE

      CVTOT=CVDR(1,1)+CVDR(3,3)+CVDR(5,5)
      CVDIFY=CVTOT-CVDY(1,1)-CVDY(3,3)-CVDY(5,5)
      CVDIFZ=CVTOT-CVDZ(1,1)-CVDZ(3,3)-CVDZ(5,5)
      DEN=CVDIFY*CVDIFY+CVDIFZ*CVDIFZ
      IF (DEN .GT. 1.0E-20) THEN
        DEN=SQRT(DEN)
        DVY=DVY*CVDIFY/DEN
        DVZ=DVZ*CVDIFZ/DEN
       ELSE
        RETURN
      ENDIF

C  ENSURE C.E. SOLUTION IS WITHIN THRUST LIMITS
      RTGO=TGO-H
      RADICL=RTGO*RTGO-(DELTAY+DELTAY)*RTGO*H/MAXDV
```

```
      IF (RADICL .LE. 0.0) THEN
        DELTAY=MAXDV*INT(RTGO/H)/2.0
        COST1=(DELTAY+DELTAY)**2
       ELSE
        COST1=((RTGO-SQRT(RADICL))*MAXDV/H)**2
      ENDIF
      RADICL=RTGO*RTGO-(DELTAZ+DELTAZ)*RTGO*H/MAXDV
      IF (RADICL .LE. 0.0) THEN
        DELTAZ=MAXDV*INT(RTGO/H)/2.0
        COST1=COST1+(DELTAZ+DELTAZ)**2
       ELSE
        COST1=COST1+((RTGO-SQRT(RADICL))*MAXDV/H)**2
      ENDIF

C  COMPUTE COST OF CERTAINTY EQUIVALENCE SOLUTION
      XDUALD(4)=XDUALD(4)-DELTAY
      XDUALD(6)=XDUALD(6)-DELTAZ
      XPDD(4)=XPDD(4  DELTAY
      XPDD(6)=XPDD(6)+DELTAZ
      TD=0.0
      DH=H
      RANGE=SQRT(XDUALD(1)*XDUALD(1)+
     +    XDUALD(3)*XDUALD(3)+XDUALD(5)*XDUALD(5))
      R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
      CALL EKF8(XDUALD,XEDD,XPDD,TD,DH,CVDD,Q,R3,
     +          0.0,0.0,0.0,KFLAG,RES,2)
      DTGO=TGO-DH
      DH=DTGO/COUNT
      DO 200 I=1,COUNT
        RANGE=SQRT(XDUALD(1)*XDUALD(1)+
     +      XDUALD(3)*XDUALD(3)+XDUALD(5)*XDUALD(5))
        R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
        CALL EKF8(XDUALD,XEDD,XPDD,TD,DH,CVDD,Q,R3,
     +            0.0,0.0,0.0,KFLAG,RES,2)
200   CONTINUE
      CVTOT=CVDD(1,1)+CVDD(3,3)+CVDD(5,5)
      COST1=COST1+K*(CVTOT+XDUALD(1)*XDUALD(1)+
     +  XDUALD(3)*XDUALD(3)+XDUALD(5)*XDUALD(5))

C  DETERMINE COST OF IMPROVING THE ESTIMATE
      XPD2(4)=XPD2(4)+DVY
      XPD2(6)=XPD2(6)+DVZ
      XDUAL2(4)=XDUAL2(4)-DVY
      XDUAL2(6)=XDUAL2(6)-DVZ

C     MOVE FORWARD ONE STEP
      DH=H
      RANGE=SQRT(XDUAL2(1)*XDUAL2(1)+
     +    XDUAL2(3)*XDUAL2(3)+XDUAL2(5)*XDUAL2(5))
      R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
      CALL EKF8(XDUAL2,XED2,XPD2,TD,DH,CVD2,Q,R3,
     +          0.0,0.0,0.0,KFLAG,RES,2)
```

```
C      COMPUTE NEW SPLINES
       CY=CY-DVY
       CZ=CZ-DVZ
       XF=((AX*TGO+BX)*TGO+CX)*TGO+DX
       YF=((AY*TGO+BY)*TGO+CY)*TGO+DY
       ZF=((AZ*TGO+BZ)*TGO+CZ)*TGO+DZ
       XFDOT=(3.0*AX*TGO+BX+BX)*TGO+CX
       YFDOT=(3.0*AY*TGO+BY+BY)*TGO+CY
       ZFDOT=(3.0*AZ*TGO+BZ+BZ)*TGO+CZ
       DTGO=TGO-H
       CALL SPLINE(XDUAL2(1),XDUAL2(2),XF,XFDOT,
      +            DTGO,AX,BX,CX,DX)
       CALL SPLINE(XDUAL2(3),XDUAL2(4),YF,YFDOT,
      +            DTGO,AY,BY,CY,DY)
       CALL SPLINE(XDUAL2(5),XDUAL2(6),ZF,ZFDOT,
      +            DTGO,AZ,BZ,CZ,DZ)

C      COMPUTE NEW CERTAINTY EQUIVALENCE SOLUTION
       CALL SEARCHA(AX,BX,CX,DX,AY,BY,CY,DY,
      +  AZ,BZ,CZ,DZ,K,DTGO,DVYA,DVZA,TOL,CFLAG)

C      ENSURE NEW C.E. SOLUTION IS WITHIN THRUST LIMITS
       RTGO=DTGO-H
       RADICL=RTGO*RTGO-(DVYA+DVYA)*RTGO*H/MAXDV
       IF (RADICL .LE. 0.0) THEN
         DVYA=MAXDV*INT(RTGO/H)/2.0
         COST2=(DVYA+DVYA)**2
        ELSE
         COST2=((RTGO-SQRT(RADICL))*MAXDV/H)**2
       ENDIF
       RADICL=RTGO*RTGO-(DVZA+DVZA)*RTGO*H/MAXDV
       IF (RADICL .LE. 0.0) THEN
         DVZA=MAXDV*INT(RTGO/H)/2.0
         COST2=COST2+(DVZA+DVZA)**2
        ELSE
         COST2=COST2+((RTGO-SQRT(RADICL))*MAXDV/H)**2
       ENDIF

C      COMPUTE NEXT ITERATION COST
       XDUAL2(4)=XDUAL2(4)-DVYA
       XDUAL2(6)=XDUAL2(6)-DVZA
       XPD2(4)=XPD2(4)+DVYA
       XPD2(6)=XPD2(6)+DVZA
       DH=DTGO/COUNT
       DO 400 I=1,COUNT
         RANGE=SQRT(XDUAL2(1)*XDUAL2(1)+
      +     XDUAL2(3)*XDUAL2(3)+XDUAL2(5)*XDUAL2(5))
         R3(1,1)=SIGMAM(1)*RANGE*SIGMAM(1)*RANGE
         CALL EKF8(XDUAL2,XED2,XPD2,TD,DH,CVD2,Q,R3,
      +            0.0,0.0,0.0,KFLAG,RES,2)
400    CONTINUE
       CVTOT=CVD2(1,1)+CVD2(3,3)+CVD2(5,5)
```

```
      COST2=COST2+K*(CVTOT+XDUAL2(1)*XDUAL2(1)+
     +    XDUAL2(3)*XDUAL2(3)+XDUAL2(5)*XDUAL2(5))+
     +    DVY*DVY+DVZ*DVZ

C  CHOOSE THE LEAST COST OPTION
      IF (COST1 .GT. COST2) THEN
        PRINT *,' DEVIATING FROM NOMINAL PATH'
        DELTAY=DVY
        DELTAZ=DVZ
        TGO=DTGO+H
      ENDIF

      END



      SUBROUTINE SEARCHT(XP,XE,A,MDOT,H,T,TGO,
     +                   DELTAY,DELTAZ,TOL,CFLAG,COUNT)

C  THIS SUBROUTINE VARIES THE TIME-TO-GO (TGO) AND THE
C  VELOCITY CHANGES (DELTAY AND DELTAZ) WHILE BRINGING
C  MISS DISTANCE WITHIN A PRESPECIFIED TOLERANCE.
C  THIS IS ACCOMPLISHED BY EMPLOYING A NEWTON-RAPHSON
C  SEARCH SCHEME FOR NON-LINEAR SYSTEMS.  (PP. 176-179
C  OF MARON, 'NUMERICAL ANALYSIS, A PRACTICAL APPROACH')
C  XF,YF,ZF,XFDOT,YFDOT, AND ZFDOT ARE COMPUTED
C  NUMERICALLY.

      REAL*8 XP(6),XE(6),XPD(6),XED(6),T,TGO,DELTAY
      REAL*8 XF,YF,ZF,SCALE,XFDOT,YFDOT,ZFDOT,TD,DTGO
      REAL*8 A,MDOT,TOL,AD,MDOTD,OLDTGO,DH,DELTAZ,H
      REAL*8 DDY,DDZ
      INTEGER I,J,COUNT,CFLAG

C  INITIALIZE VELOCITY CHANGES
      DELTAY=0.0
      DELTAZ=0.0
      OLDTGO=TGO

C  BEGIN SEARCH LOOP
      DO 10 J=1,10

C  INITIALIZE DUMMY VARIABLES
        DO 110 I=1,6
          XED(I)=XE(I)
          XPD(I)=XP(I)
110     CONTINUE
        XPD(4)=XPD(4)+DELTAY
        XPD(6)=XPD(6)+DELTAZ
        AD=A
        MDOTD=MDOT
```

```
C  PROPAGATE DUMMY VARIABLES FORWARD TO IMPACT TIME
         TD=T
         DH=TGO/COUNT
         DO 115 I=1,COUNT
           CALL RK4SYSP(TD,XPD,DH)
           CALL RK4SYSE(TD,XED,DH,AD,MDOTD)
           TD=TD+H
115      CONTINUE

C  ASSIGN FINAL RELATIVE POSITIONS
         XF=XED(1)-XPD(1)
         YF=XED(3)-XPD(3)
         ZF=XED(5)-XPD(5)
         MISS=SQRT(XF*XF+YF*YF+ZF*ZF)

C  TEST FOR NEARNESS
         IF (MISS .LE. TOL) RETURN

C  ASSIGN FINAL RELATIVE VELOCITIES
         XFDOT=XED(2)-XPD(2)
         YFDOT=XED(4)-XPD(4)
         ZFDOT=XED(6)-XPD(6)

C  COMPUTE CHANGES IN CONTROL PARAMETERS AND UPDATE
         DTGO=-XF/XFDOT
         DDY=(DTGO*YFDOT+YF)/TGO
         DDZ=(DTGO*ZFDOT+ZF)/TGO
         TGO=TGO+DTGO
         DELTAY=DELTAY+DDY
         DELTAZ=DELTAZ+DDZ

C  TEST FOR CONVERGENCE OF TIME-TO-GO
         IF (ABS(DTGO) .LT. (.000001*TGO)) RETURN

10     CONTINUE

C  INCREMENT CONVERGENCE FLAG
       CFLAG=CFLAG+1

C  ZERO OUT VELOCITY CHANGES AND RESET TIME
       DELTAY=0.0
       DELTAZ=0.0
       TGO=OLDTGO

       END
```

APPENDIX E

IN-PLANE THRUST PROFILES

This appendix contains the in-plane thrust profiles for Cases I and V, showing the effect of estimate uncertainty on the various control strategies. Each profile was started with the same random seed to form a basis for comparison.

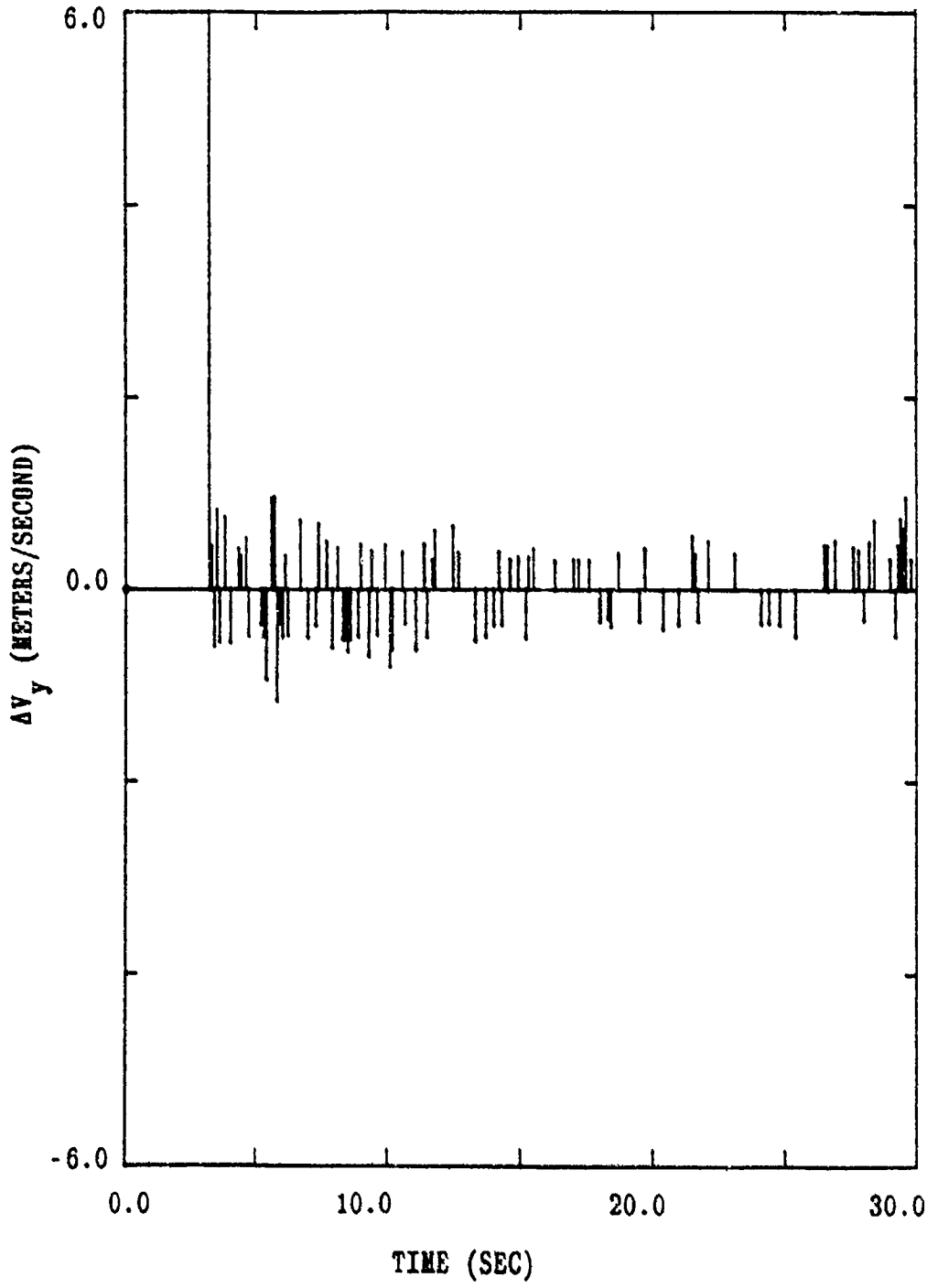

Figure E-1. In-plane thrust profile of Plan A for Case I.

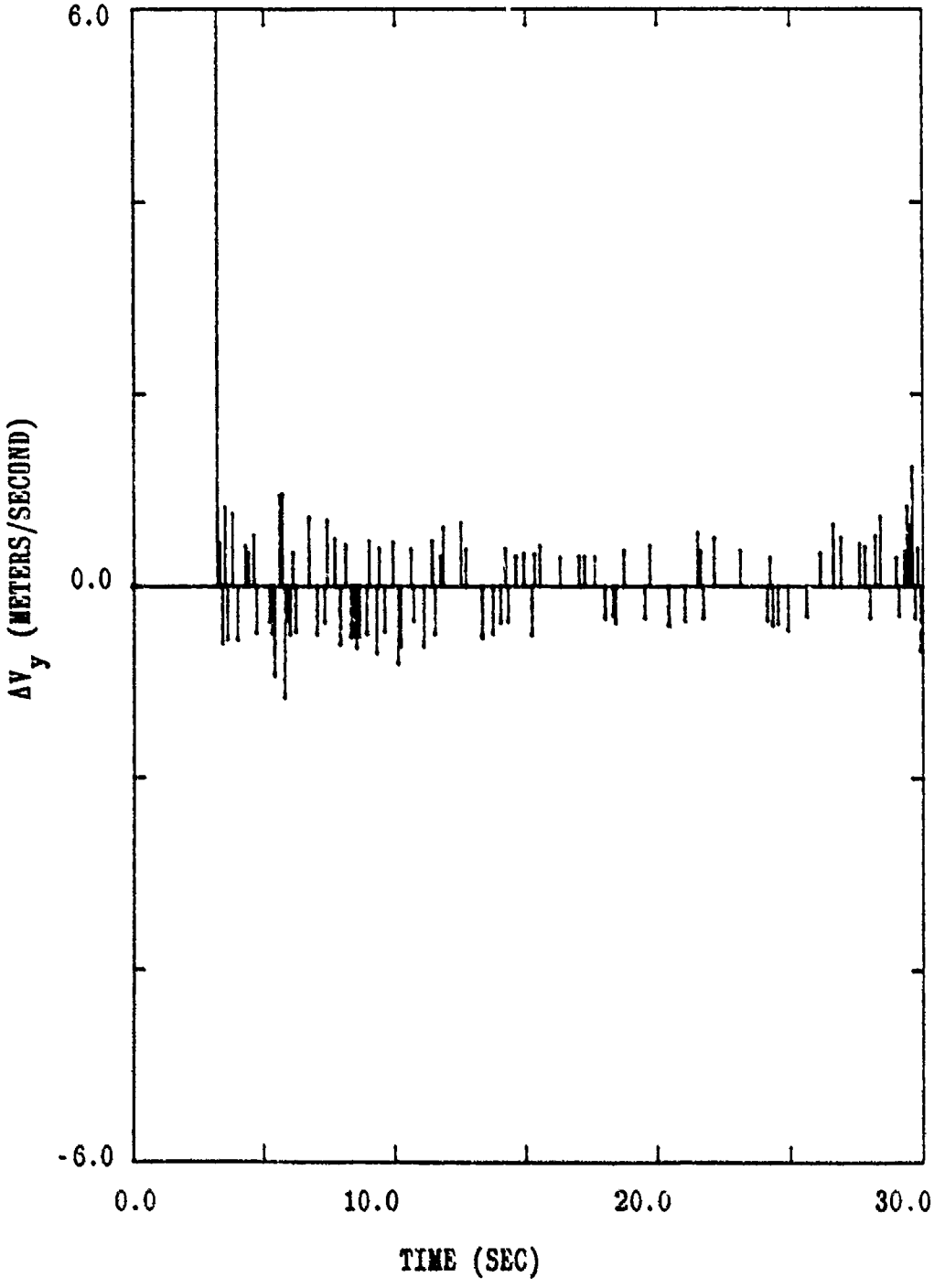

Figure E-2. In-plane thrust profile of Plan B for Case I.

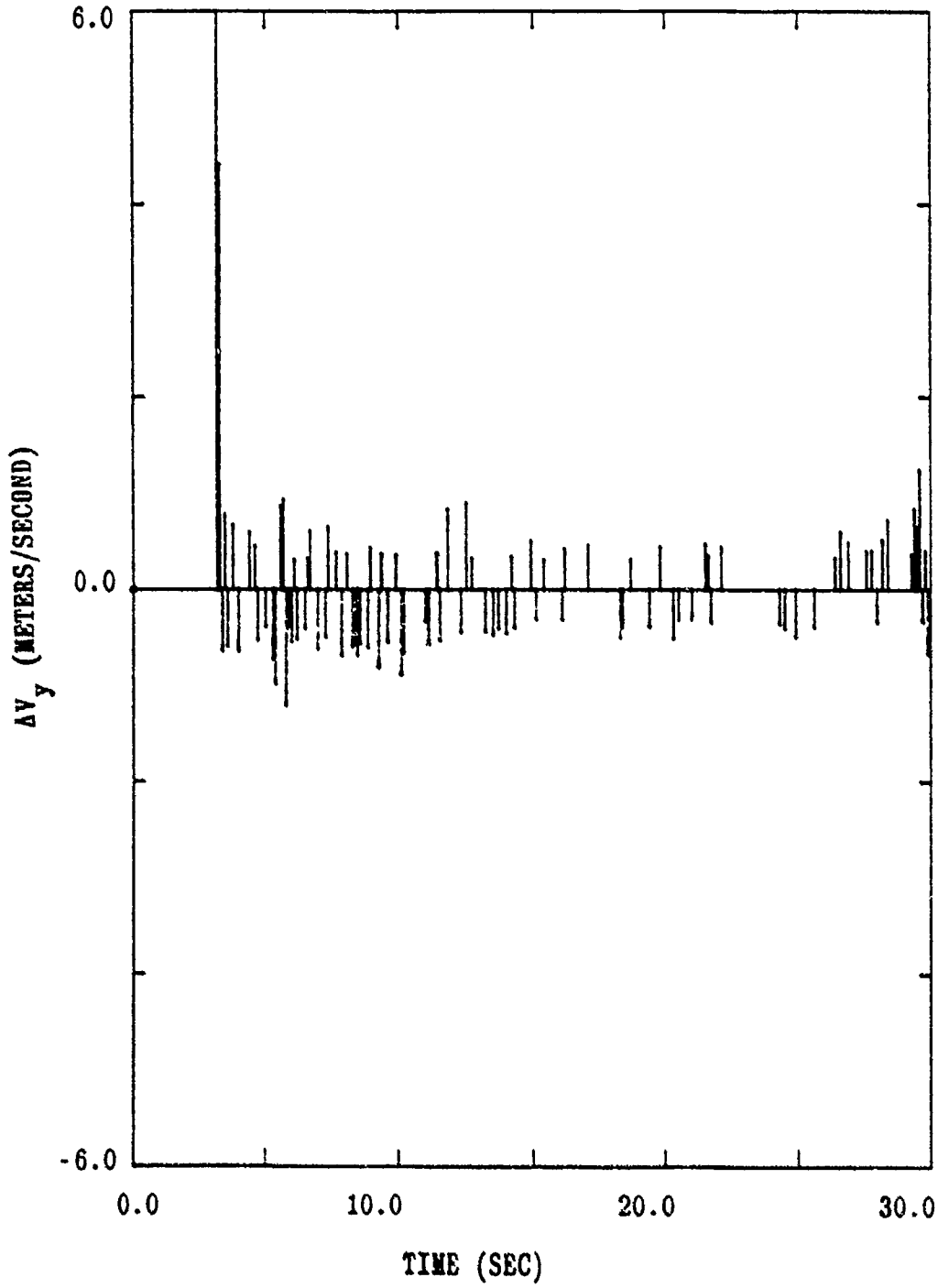

Figure E-3. In-plane thrust profile of Plan C for Case I.

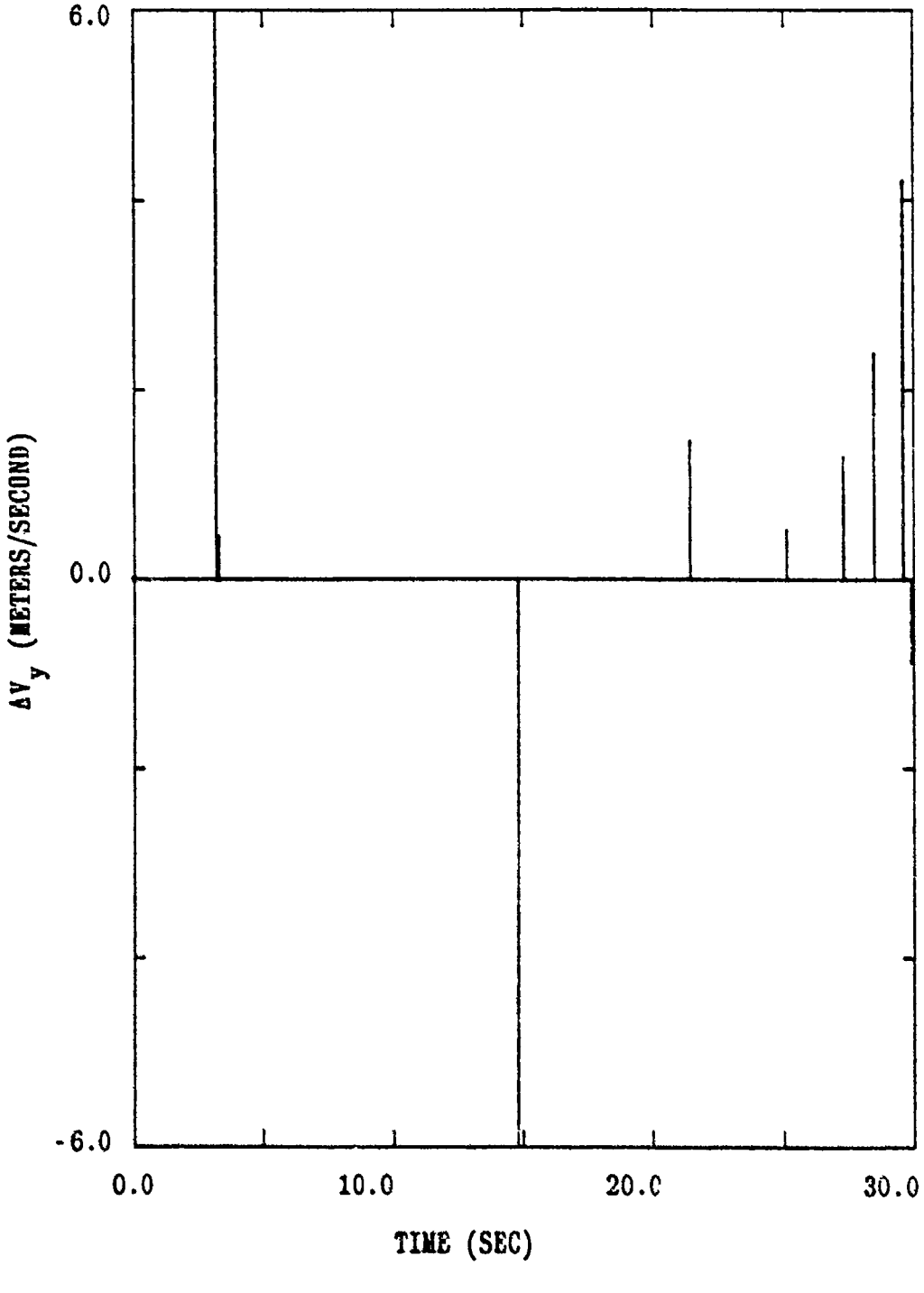

Figure E-4. In-plane Optimium Thrust Spacing profile for Case I.

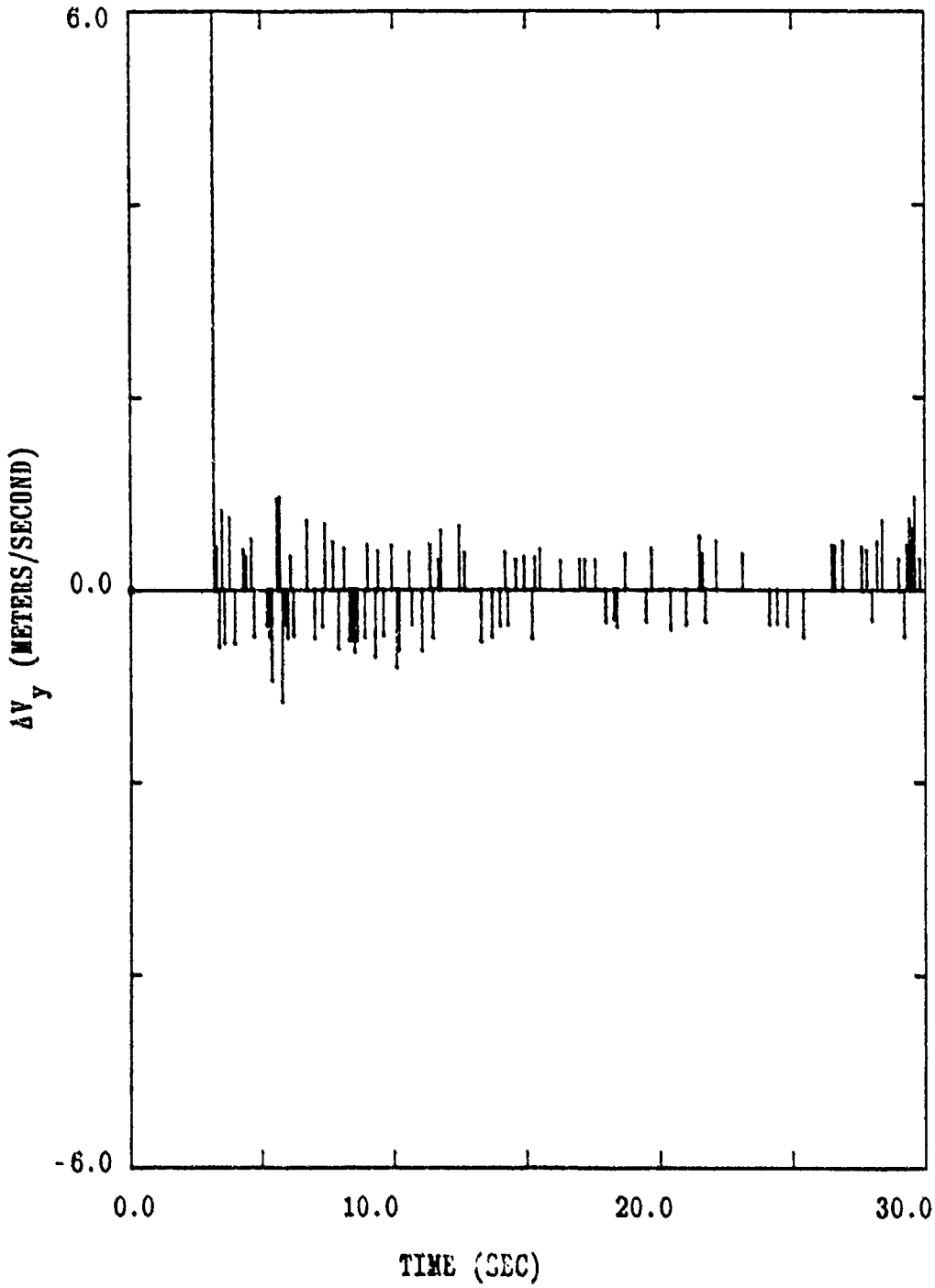

Figure E-5. In-plane thrust profile of Dual Control for Case I.

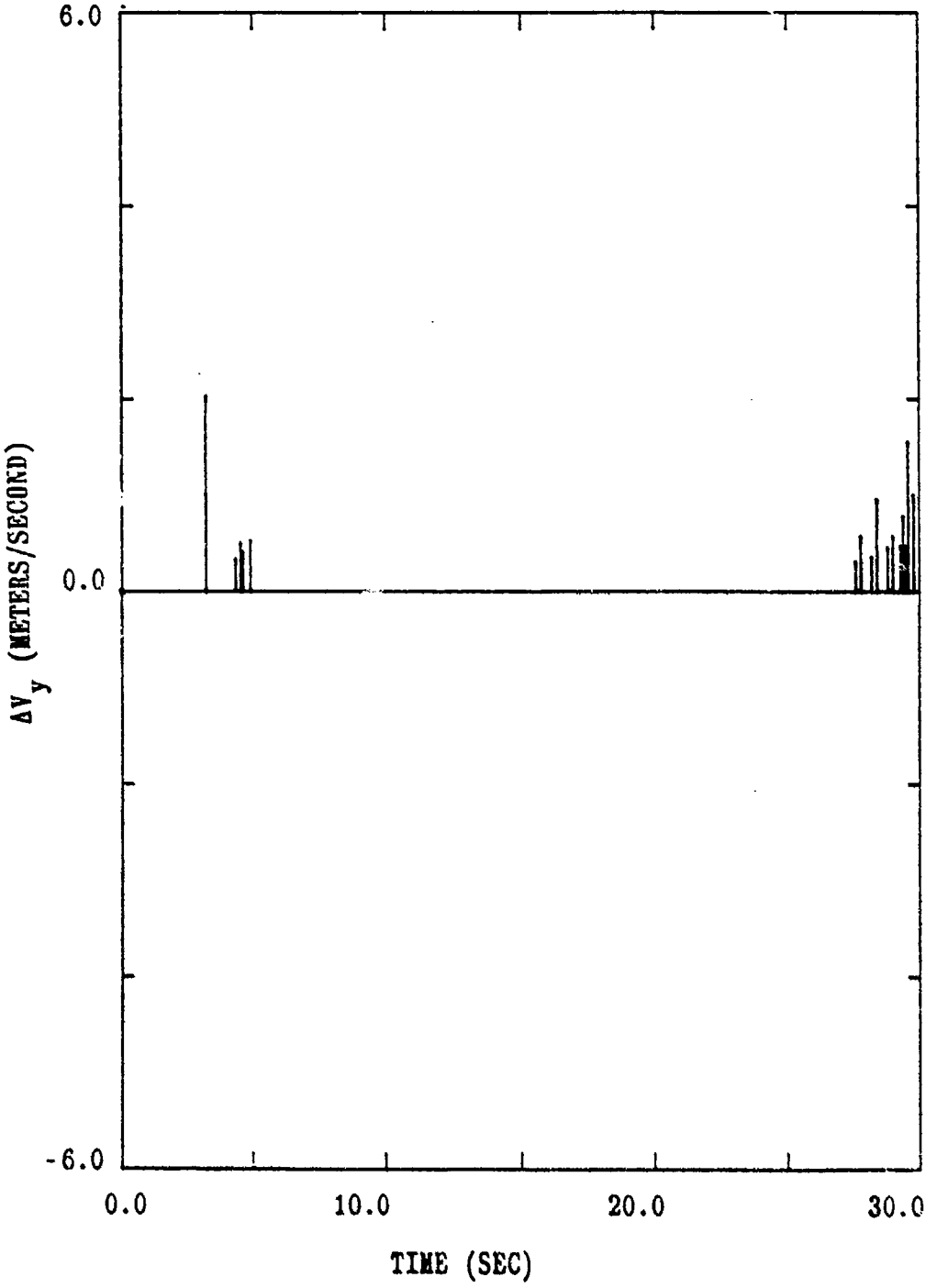

Figure E-6. In-plane thrust profile of Certainty Control for Case I.

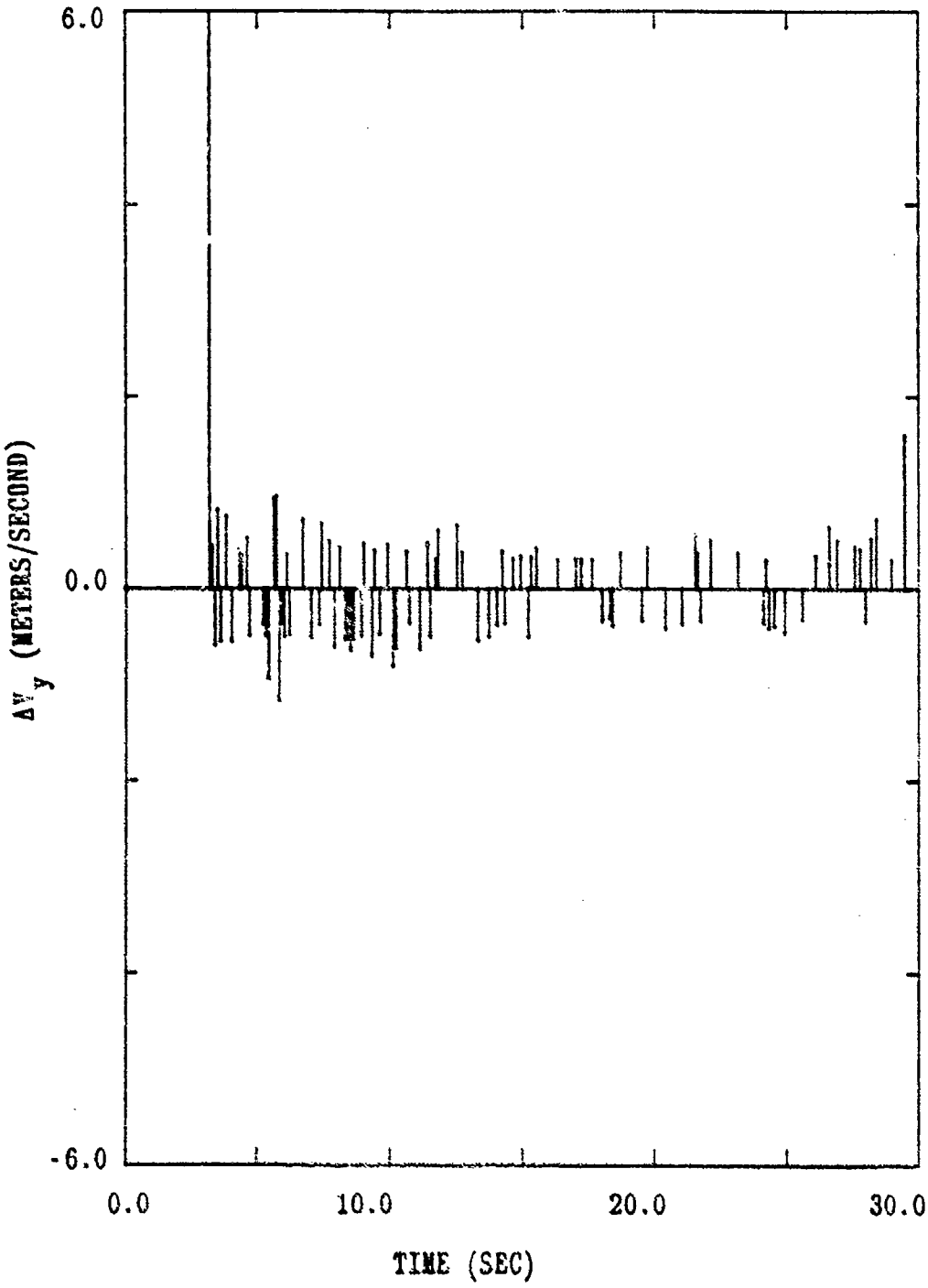

Figure E-7. In-plane thrust profile of Truth Model for Case I.

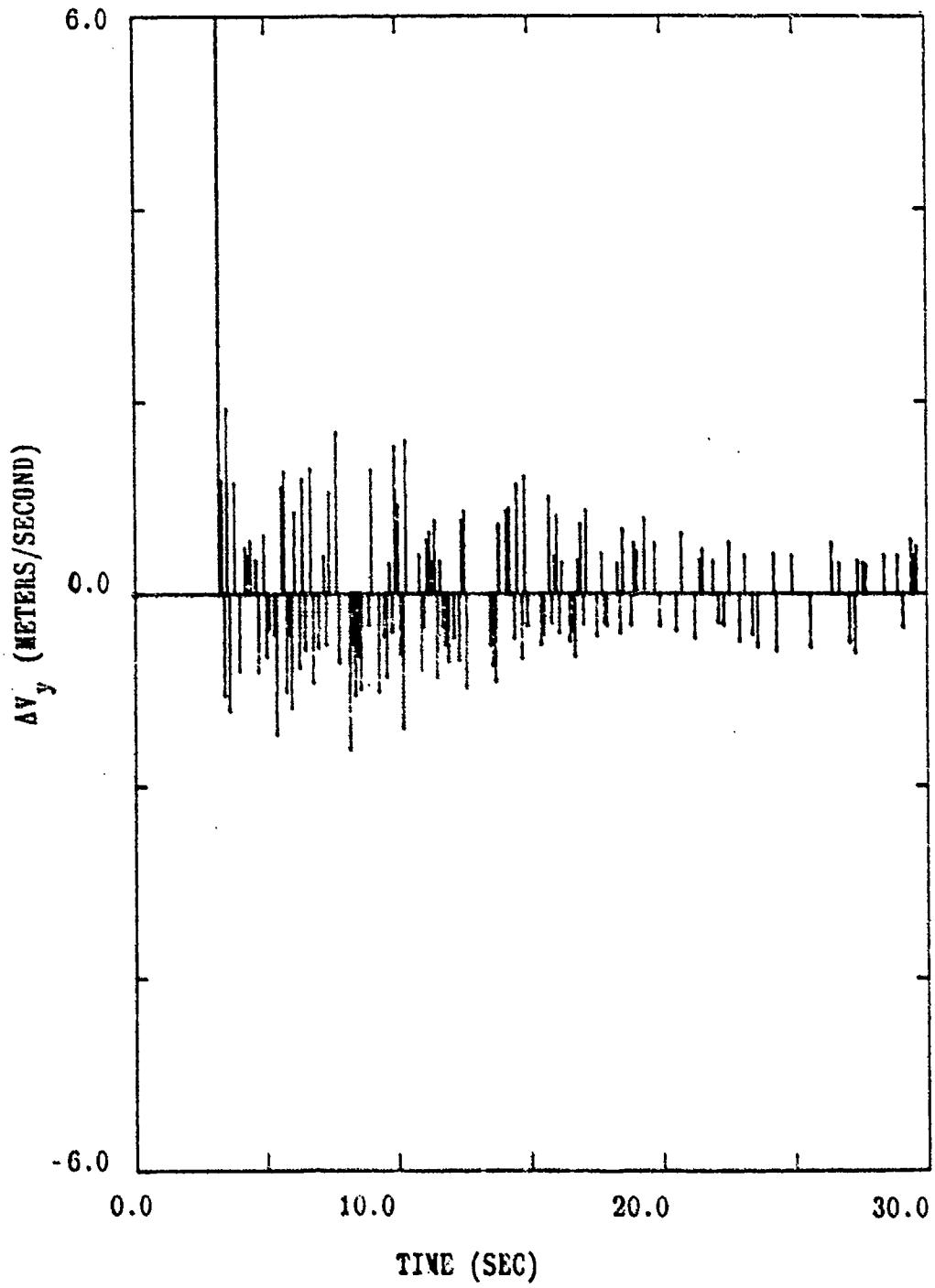

Figure E-8. In-plane thrust profile of Plan A for Case V.

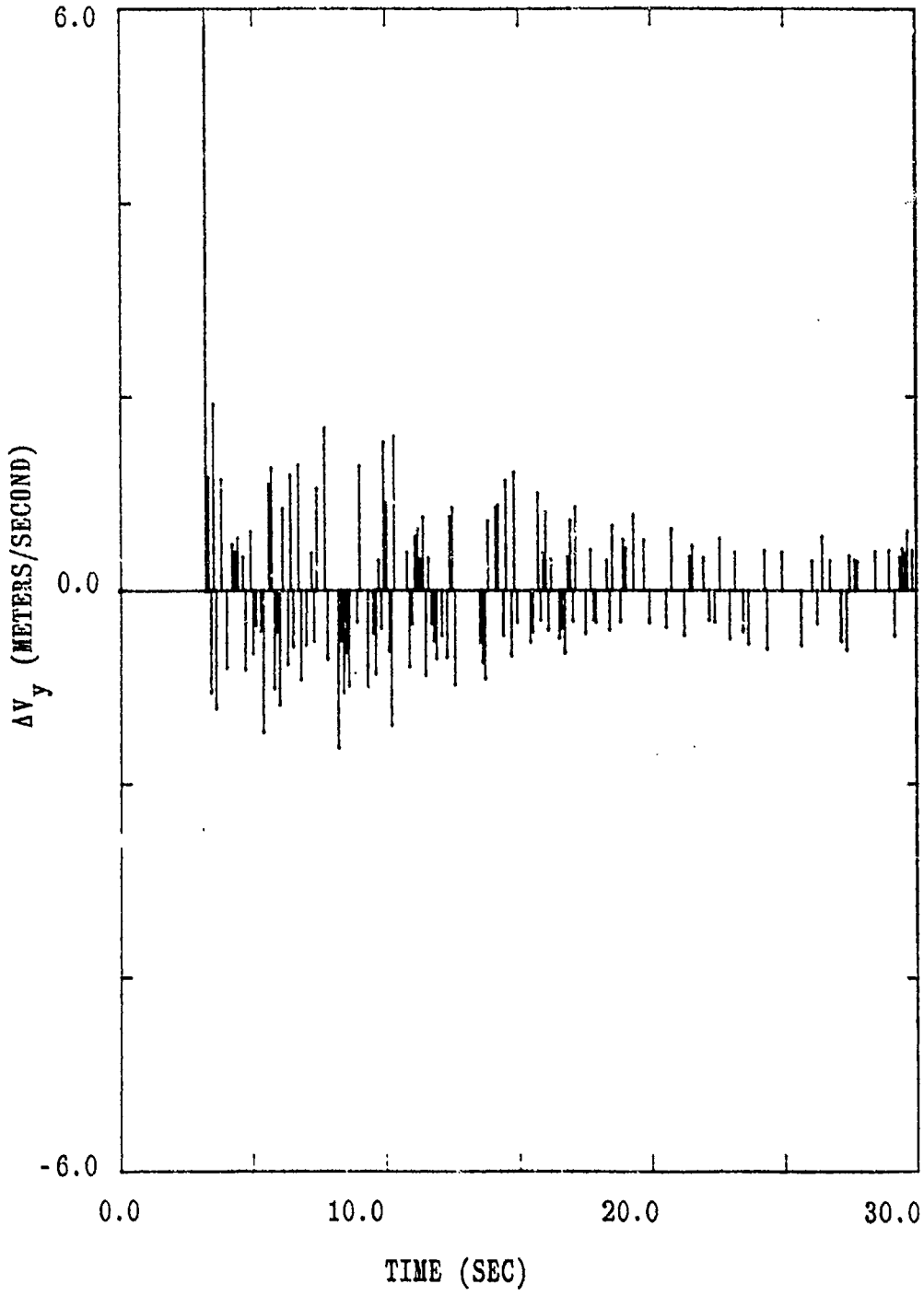

Figure E-9. In-plane thrust profile of Plan B for Case V.

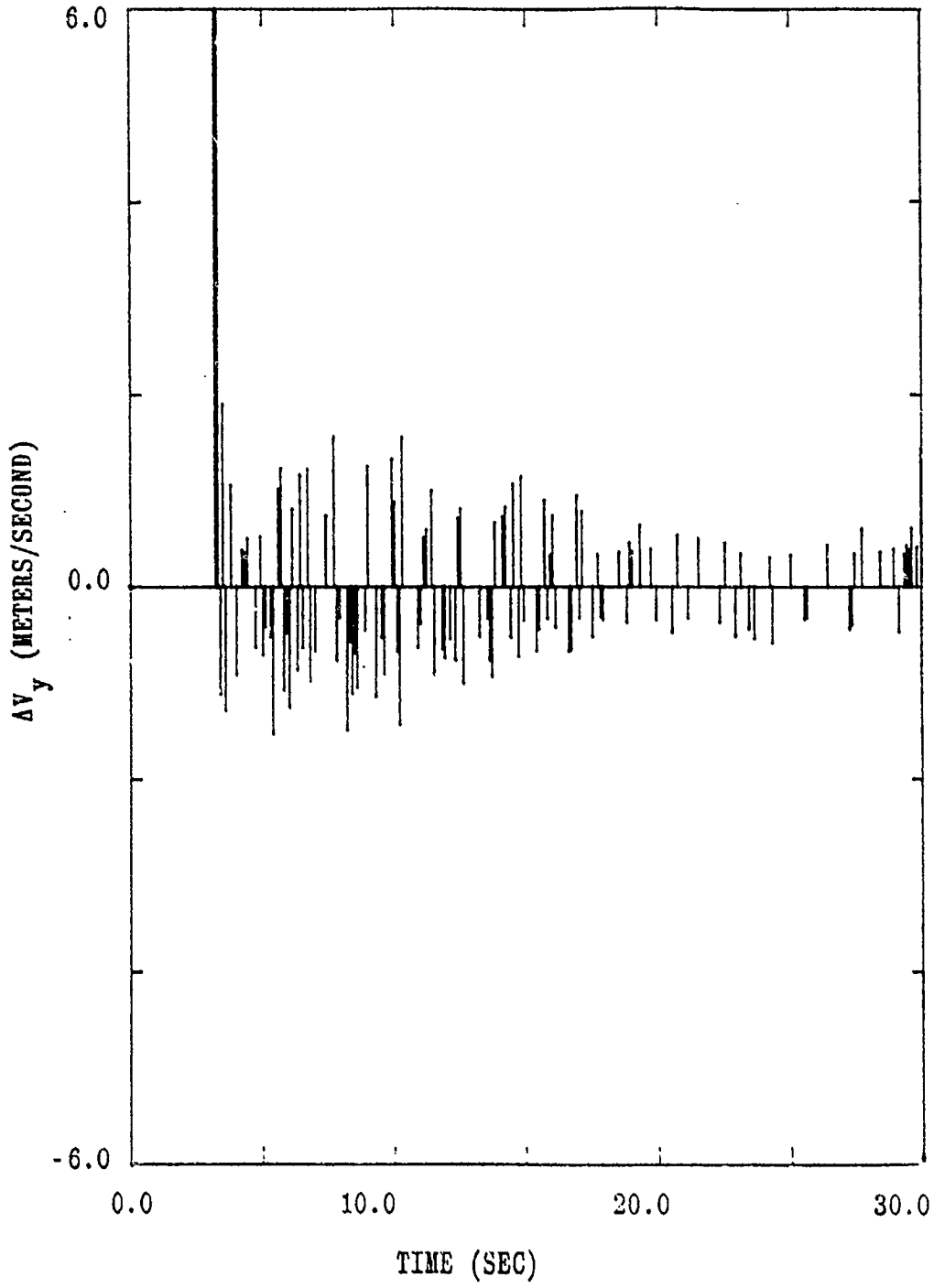

Figure E-10. In-plane thrust profile of Plan C for Case V.

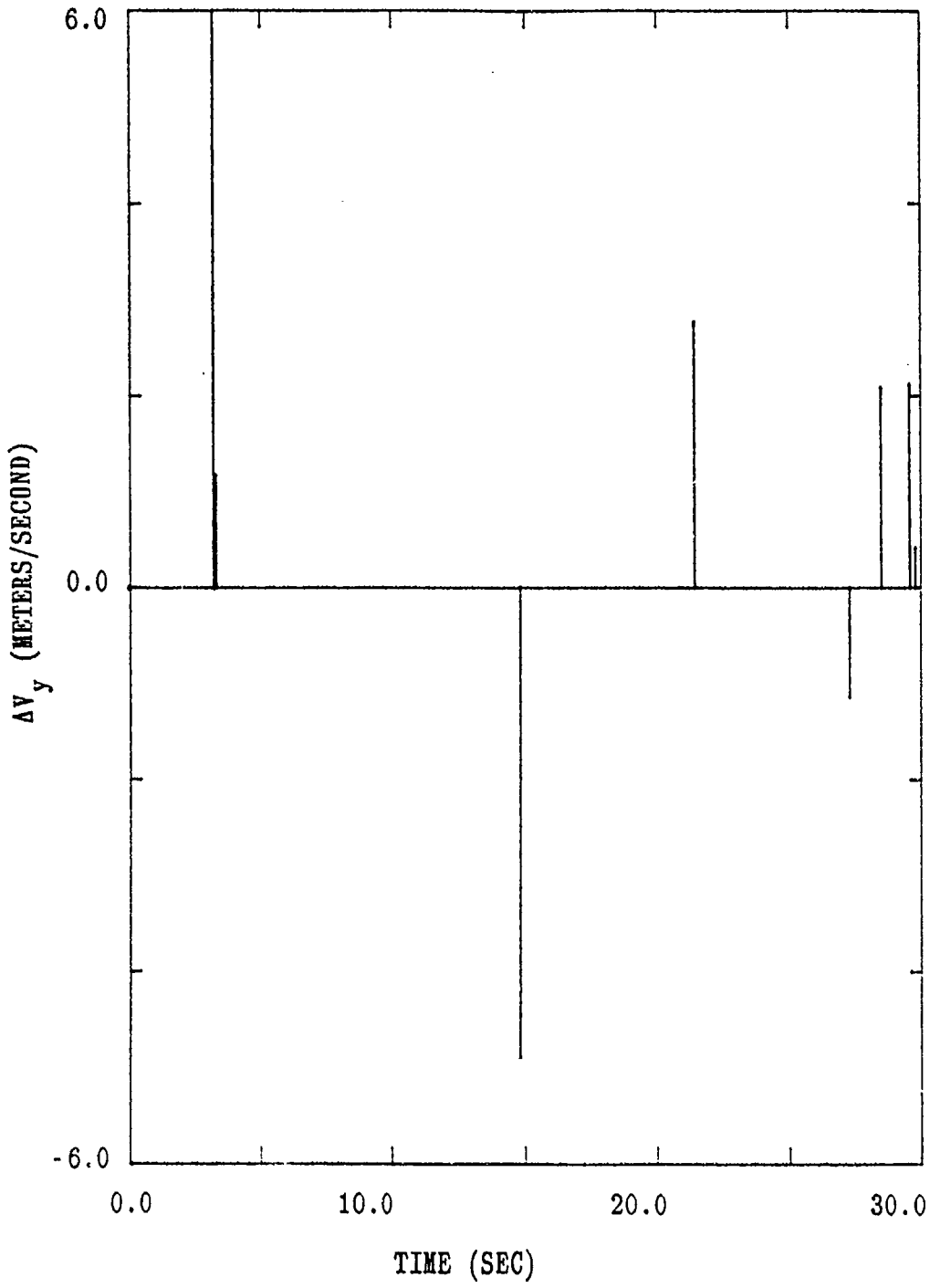

Figure E-11. In-plane Optimium Thrust Spacing profile for Case V.

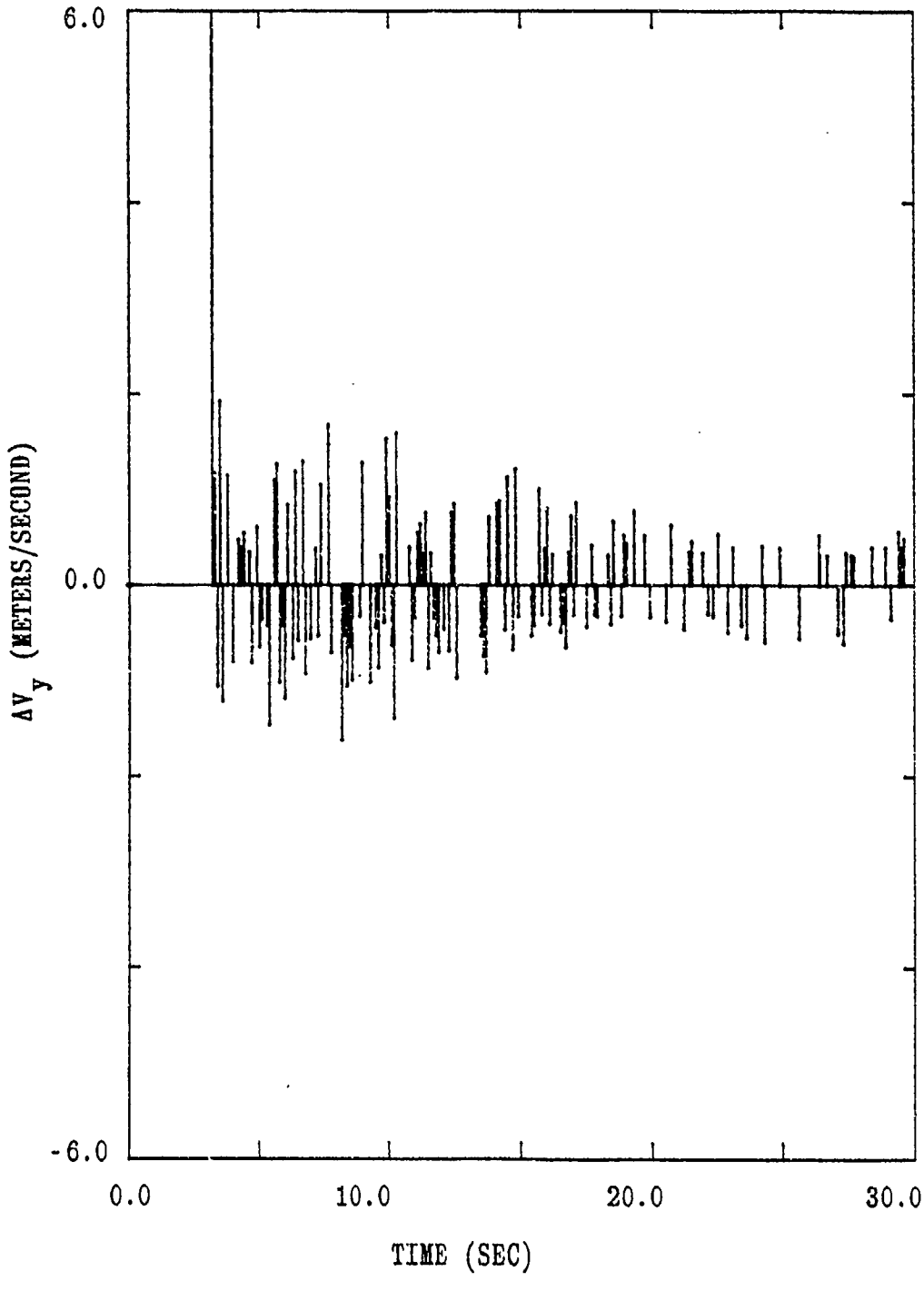

Figure E-12. In-plane thrust profile of Dual Control for Case V.

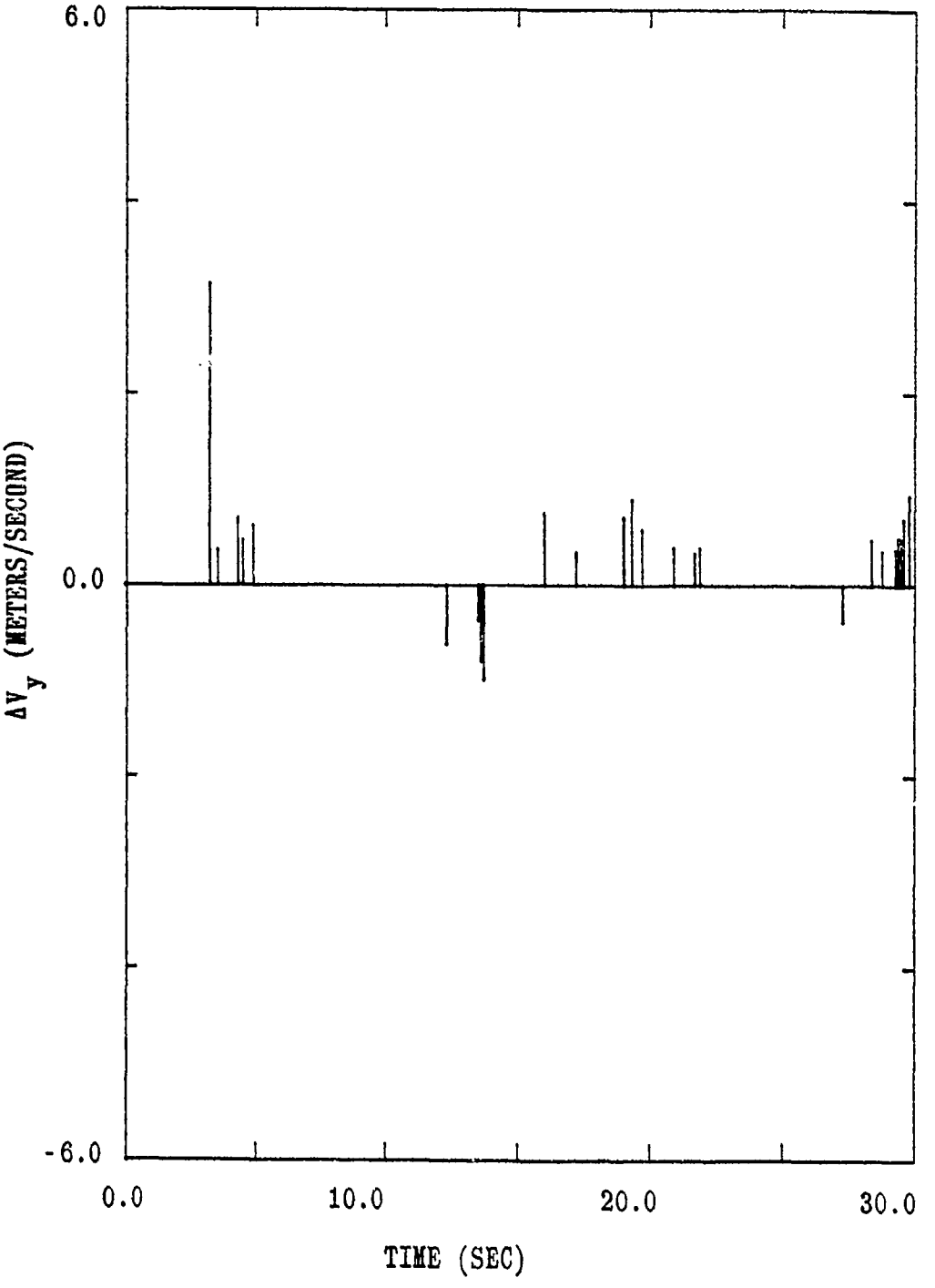

Figure E-13. In-plane thrust profile of Certainty Control for Case V.

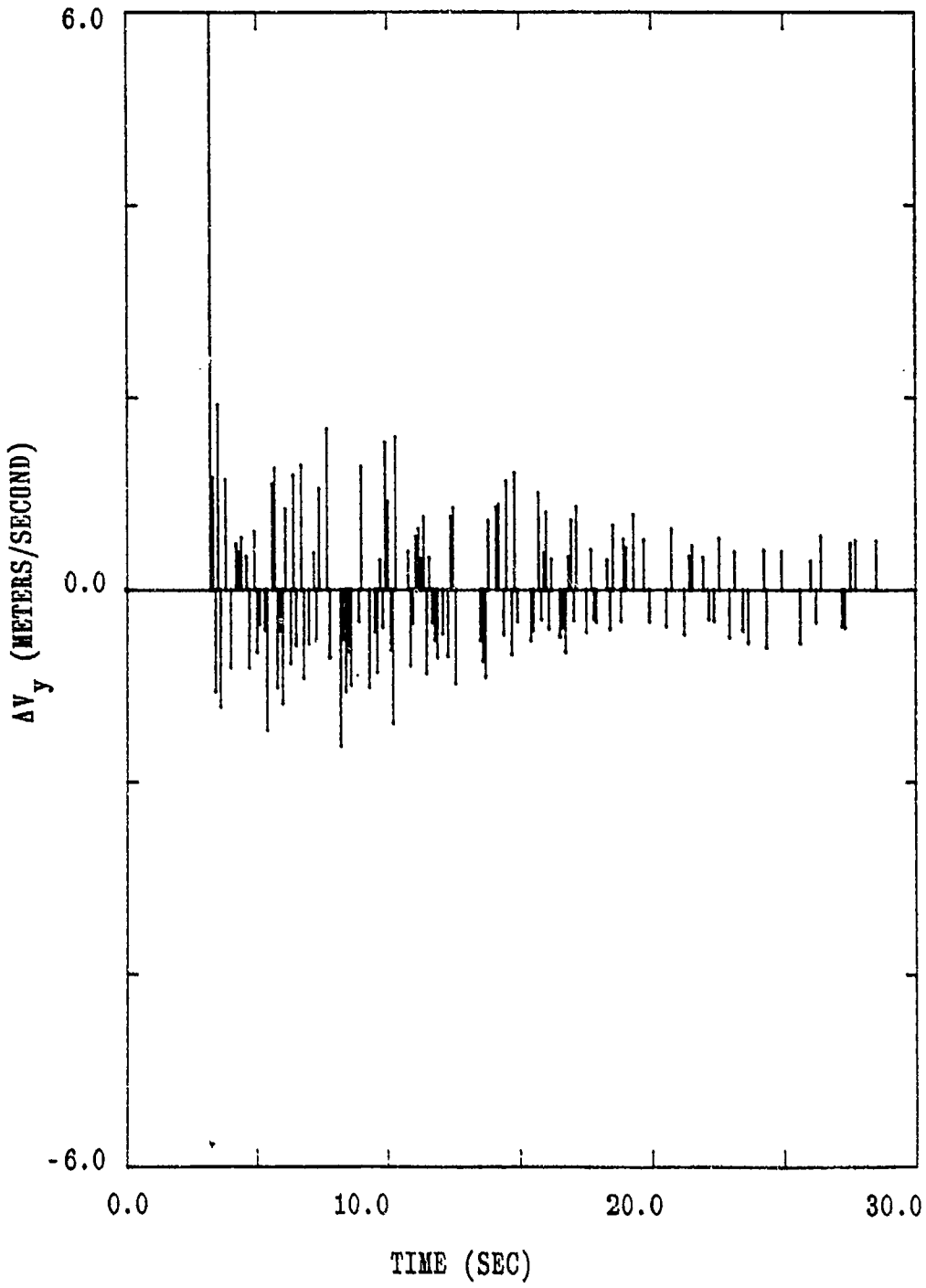

Figure E-14. In-plane thrust profile of Truth Model for Case V.

NAME: Salvatore Alfano, Major, USAF

TITLE: Hypervelocity Orbital Intercept Guidance

DEGREE: Ph.D., Electrical Engineering, 1988,
University of Colorado at Colorado Springs.

NUMBER OF PAGES: 183

ABSTRACT: Terminal guidance of a hypervelocity exo-atmospheric orbital interceptor with free end-time is examined. The pursuer is constrained to lateral thrusting with the evader modeled as an ICBM in its final boost phase. Proportional navigation, optimal control using certainty equivalence, dual control, and control with optimum thrust spacing are all examined. Also, a new approach called certainty control is developed for this problem. This algorithm constrains the final state to a function of projected estimate error to reduce control energy expenditure. All methods model the trajectories using splines and employ eight state Extended Kalman Filters with line-of-sight and range updates. The relative effectiveness of these control strategies is illustrated by applying them to various intercept problems.